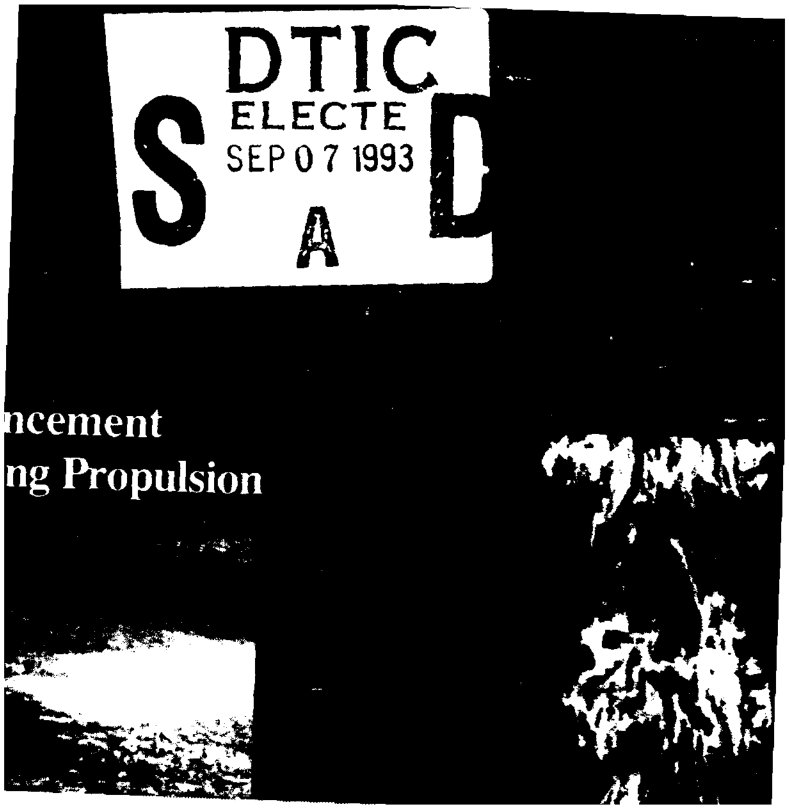
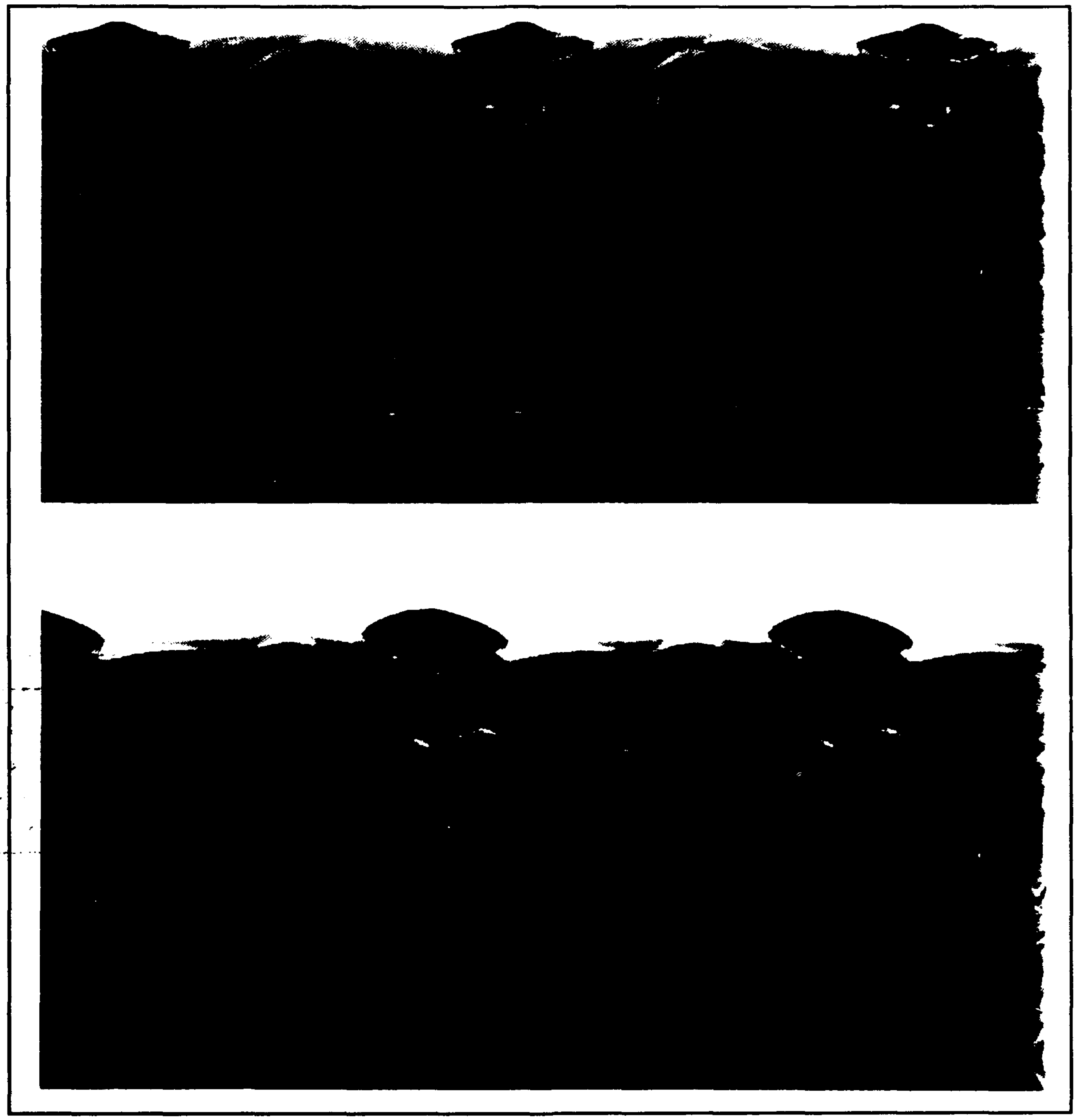

ncement
ng Propulsion



The above pictures show the analytical and computational aspects of turbulence and combustion. These figures show an unstable detonation wave in two distinct states. The top figure shows the temperature profile of a Z-N-D detonation wave with q_0 (the heat release parameter) of 50, E^+ (the activation energy) of 10, and f (the overdriver of the detonation) of 1.2. Note the generation of regularly paired eddies. The bottom picture is produced under the same parameter values except for E^+ that has a value of 50. This picture shows the generation of strong irregular cells and fully developed 2-D turbulence.

This pioneering research of Professor Andrew Majda and co-workers at Princeton University was sponsored by the Office of Naval Research. On page 20, Professor Majda is featured in "Profiles in Science."

ONR has been working in this field of research for 30 years. The discovery in the 1960's under ONR support of coherence in turbulence revolutionized the view of turbulent flow. Before that time the statistical description of turbulence emphasized its randomness and brought about time-averaged turbulence models. ONR-sponsored research identified coherent structures in both free and bounded shear flows, and subsequent programs investigated the role of these structures in the production and dissipation of energy in the flow.

Naval Research Reviews

Office of Naval Research
Four / 1992 One / 1993
Vol XLIV / Vol XLV

Articles

Departments

About the Cover

Increasing, mixing, and generating the optimum large scale structures and small scale turbulence are crucial to increase the combustion efficiency of ramjets (photograph on the left hand side of the cover). Non axisymmetric nozzles have been found to generate enhanced mixing by the phenomena called "axis switching," where the vortex rings deform to shift axes (major axis to minor axis). This causes the jet to spread more, generating increased mixing. On the right hand side is shown a square jet with the deformed vortex rings at the bottom, and the longitudinal vortices and their merging at the top.

Naval Research Reviews publishes articles about research conducted by the laboratories and contractors of the Office of Naval Research and describes important naval experimental activities. Manuscripts submitted for publication, correspondence concerning prospective articles, and changes of address, should be directed to Code OPARI, Office of Naval Research, Arlington, VA 22217-5000. ~~Requests for subscriptions should be directed to the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20403~~. *Naval Research Reviews* is published from appropriated funds by authority of the Office of Naval Research in accordance with Navy Publications and Printing Regulations. NAVSO P-35.

Research on Multiple-Valued Logic At The Naval Postgraduate School

Jon T. Butler
Department of Electrical and Computer Engineering
Naval Postgraduate School

Abstract

The Navy's need for faster, smaller computers has inspired research on the use of more than two levels of logic in computer circuits. This paper focusses on work at NPS in multiple-valued logic. It discusses the development of multiple-valued programmable logic arrays (MVL-PLA), as well as computer-aided design tools for MVL-PLA's.

Introduction

The effectiveness of Navy systems is dependent on computers. Computers are used to predict weather, to track resources, to guide missiles, to simulate combat, and to process the words and numbers necessary for the general conduct of Navy business. Their usefulness depends on computing power, speed, and size. As with other critical resources, the Navy cannot depend on the private sector as the exclusive provider of the technology; it must be an active participant. This article focuses on computer circuits, and specifically on multiple-valued circuits, whose use improves computing power, speed, and size. We first consider binary circuits. Then, we discuss multiple-valued circuits. This serves as introduction to the main discussion, research at the Naval Postgraduate School on this important subject.

Disadvantages of Binary

Current computers are binary; they use two values of logic. For convenience, we represent these as 0 and 1. But 0 and 1 are simply symbols for some quantity, such as voltage, current, or charge. For example, in the most widely used circuits, 0 represents 0 volts and 1 represents 5 volts.

Binary is commonly used for historical reasons. Computers in the 1940's used relays, which had two stable states, open and closed. Tubes and transistors have two stable states, saturation (conducting) and cutoff (nonconducting). There are disadvantages to binary, however. One is evident by the number of bits needed to represent a number. For example, the decimal number 33 corresponds to 10001 in binary. That is, while five symbols are needed to represent 33 in binary, only two are needed in decimal. Unfortunately, the difference increases with the size of the number. This inherent complexity is hidden from most users by hardware (e.g., binary to decimal converters in digital watches) and by software (e.g., high level languages). However, certain disadvantages cannot be so simply hidden. Real penalties exist in both compactness and speed.

Compactness

The complexity of the circuits that perform addition, multiplication, and other arithmetic operations depends on the number of symbols used. For commonly used number systems, this complexity increases with the number of symbols. As a result, so does the chip area needed for devices. However, the most significant penalty is not with the devices, but the interconnect between devices. In VLSI circuits, about 70% of the chip area is used for interconnect, 20% for insulation, and 10% for devices. Interconnect corresponds to the chip area used to carry information around the circuit. Insulation is needed to separate devices and interconnect. With so much area used to connect devices, less area is available for logic.

The inefficient use of interconnect by binary also occurs outside the chip. Signals going into and out of a VLSI chip must use pins connecting the chip to the circuit board. This is called the *pinout problem*. That is, while significant reductions have occurred in the size of logic devices, there has been no comparable reduction in the size of integrated circuit pin connections. Strength and reliability dictate a (relatively large) minimum pin connection size. As a result, there is a need to better use existing pins.

Speed

A limit on the speed of arithmetic circuits is the carry (or borrow) between digits. For example, in binary addition, the most significant sum bit depends not only on the most significant bits of the two numbers being added, but also on all lower bits (contrast this to the least significant sum bit which does not depend on higher order bits). The dependence occurs through the carry between bits. Thus, computation of the most significant sum bit must wait until the carries out of all lower order bits are computed. Addition, when done in this way, cannot be performed at high speed.

However, there are multiple-valued number systems where this dependence does not exist. An example is the residue number system. In this system, operations that form the sum, difference, or product occur only at each digit independent of all other digits. Because of this independence, arithmetic operations are fast in the residue number system.

Figure 1.

Photomicrograph of a Basic Charge Coupled Device Programmable Logic Array (Courtesy of the University of Twente, ©IEEE 1988).

Multiple-Valued Logic

All of the disadvantages discussed above can be alleviated by using more than two levels of logic. The problem of a large number of bits needed to represent a number decreases as the number of logic values increases. Interconnect, both internal and external to the VLSI chip, is more efficiently used in multiple-valued logic. The carry propagation problem disappears with the proper choice of multiple-valued number system.

However, there remains the question of implementing a multiple-valued system. Modern computing devices naturally allow more than two levels of logic. For example, in charge-coupled devices (CCD), logic levels are represented as various quantities of charge[2]. Resonant-tunnelling devices[3] allow four or more levels of voltage. Fuzzy logic devices offer large numbers of logic levels; so many, that they are modelled as infinite; present circuit implementations exceed 15 levels[17]. Bio-molecular devices, where information processing occurs at the molecule level[6], operate on an enormous number of logic levels, conservatively estimated to be over 10,000.

Multiple-Valued Programmable Logic Arrays

The design of logical circuits is a complex process. The vast amount of circuitry needed for even a simple operation like addition can be daunting. Because of this, a way is needed to organize circuits. The programmable logic array or PLA is a circuit structure designed to handle this complexity. PLA's are uniform and thus easily replicated in VLSI. They can be programmed, and therefore adapted to realize a given design specification. Because of this characteristic, our research at the Naval Postgraduate School has concentrated on PLA design.

In a cooperative venture with the University of Twente in Enschede, The Netherlands, we have developed a PLA circuit for charge-coupled device (CCD) technology (See Figure 1 from[2]). In CCD, values are stored in cells (capacitors) as charge. A logic 0, 1, 2, and 3 is stored as 0, 1,000,000, 2,000,000, and 3,000,000 electrons. Computation occurs as charge moves among cells combining and breaking up. An advantage of CCD is compactness, more so than any other VLSI technology. While slower than the common CMOS technology, it is faster than disk and stands as a replacement for disk. The use of multiple-valued logic in CCD increases its density significantly. CCD is used as a light receptor, and the advent of logic circuits in CCD means that image processing can be done on the same chip in which the image is received. With large numbers of logic levels, there is no need for encoders between light receptors and the logic used to process and store the image. CCD circuits with many logic levels have been fabricated. For example, Hitachi has succeeded in building a 16 level CCD memory.

While the uniform structure of a PLA helps one to handle the high complexity of logic circuits, there is the problem of designing large PLA's. To make multiple-valued PLA design feasible for large systems, we initiated a program to develop computer-aided design tools.

Computer-Aided Design for Multiple-Valued Logic

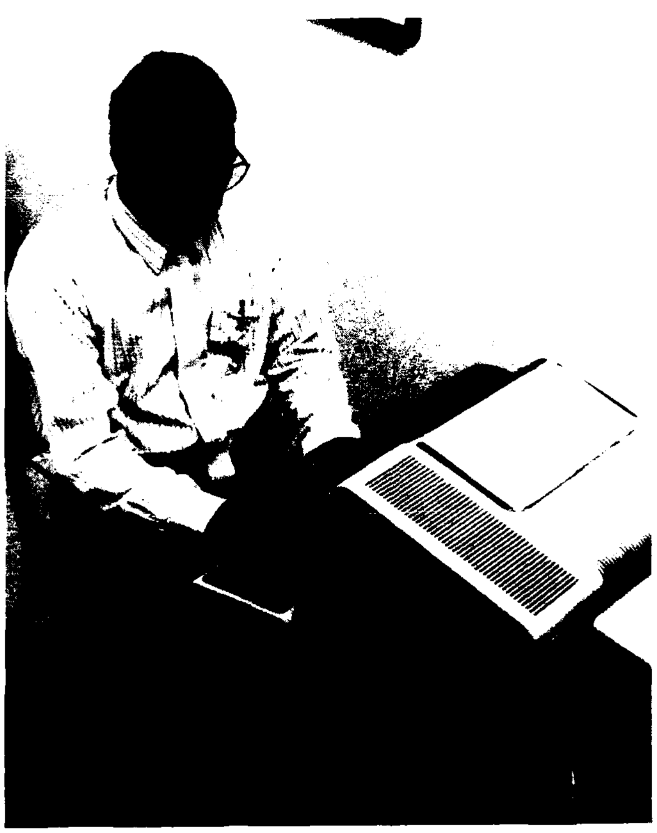
We began this program with a study of heuristic techniques for finding *minimal* PLA's, PLA's with the smallest structure. A heuristic will produce a design, although not necessarily a minimal one. In 1987 at the beginning of this study, there were three heuristic techniques. At that time, there was no systematic analysis of their relative merits; each existed separately. In an effort to gain an understanding of the multiple-valued PLA design process, we compared the three heuristics on the same functions (specifications). One interesting result[13] was that heuristics tended to perform better on different classes of functions, and there was a surprising advantage to applying all three and choosing the most compact realization.

We also provided a partial answer to an important question; How well do heuristics perform compared to the best that one could possibly do? To answer this, we developed the first practical approach to finding provably minimal solutions; an efficient search technique that is much faster than exhaustive search, but still guaranteeing a minimal solution. Indeed, without the improved search technique, we would have been unable to achieve the minimal solution because of the enormous computation time required by exhaustive search. This comparison provided much needed insight into heuristic methods and provided realism in our expectation of the quality of solutions provided by heuristic methods.

It is important to state that these experiments deal with small functions; for even medium size functions this experiment would have been impossible. For larger functions, the only approach is heuristic. Indeed, our research shows that often a minimal realization is *not* produced. Although a designer will not know how close his/her design is to the optimum, our study indicates that deviations are probably less than 10% away. Ours was the first systematic study of the problem.

Figure 2.

LTJG Ugur Ozkan Using HAMLET.



Our experience from this analysis showed the need for a tool to analyze PLA design heuristics. In 1988, we launched an extensive design project for such a tool. The result was HAMLET (Heuristic Analyzer for Multiple-Valued Logic Expression Translation), the first practical computer-aided (CAD) tool for multiple-valued logic. HAMLET[16] has the ability to analyze heuristics specified by the user, using either individual functions or randomly generated functions. This is a particularly useful feature. Instead of writing a separate program to generate test inputs, this is done automatically in HAMLET. Of special significance is HAMLET's ability to collect and automatically display heuristics. Tasks that normally took man-days can now be literally done in keystrokes. HAMLET can produce ensembles of test cases whose statistical properties can be controlled. The search algorithm used in reference 13 was also incorporated into HAMLET. We are now able to use the search to find solutions that were impossible to obtain with any previous method. The search can be controlled. At one extreme, it can go directly to a PLA realization of a given function. At the other, it can perform a complete search, producing a known minimal solution. In effect, this is like a dial, one end of which corresponds to fast but poor solutions, while the other end corresponds to slow but minimal solutions. HAMLET can also do design, producing the layout of a PLA, given a specification. Figure 3 shows one PLA layout produced by HAMLET. HAMLET is designed to have a long lifetime. It is easily modified; indeed an improved heuristic has been added[14], as well as an entirely new approach[4] (see below). Also, HAMLET has been used by researchers in at least four countries.

Simulated Annealing

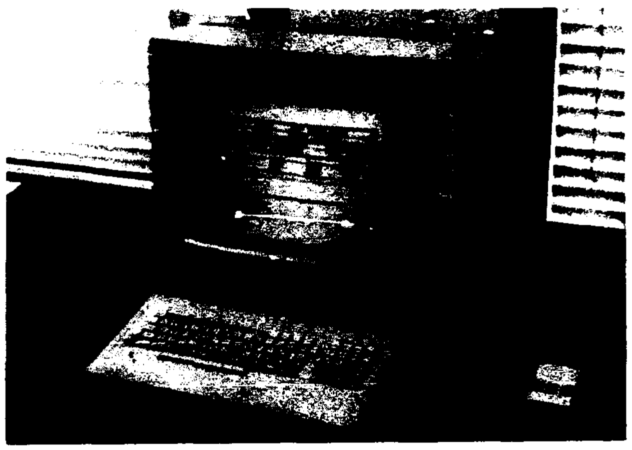
When metal or glass is heated and then slowly cooled, it is said to undergo *annealing*. The final cooled state is similar to a crystal and represents a low energy state. If it is cooled at a high rate, atoms have less of a chance to align and a higher energy state occurs.

In 1953, Metropolis[11] proposed a method of optimizing complex problems that mimics this process. It is called *simulated annealing*. We have formulated simulated annealing for the PLA minimization problem as follows. One can satisfy a given PLA specification by producing a set of implicants. Each implicant represents a part of the whole PLA. Solving the PLA minimization problem, therefore, is equivalent to finding the fewest implicants that satisfies the specification. Indeed, there is a direct relationship between the number of implicants and the chip area occupied. Thus, reducing the number of implicants by half, for example, results in almost a one-half reduction in chip area. Implicants can be manipulated. For example, one implicant can be divided into two, pairs sometimes can be combined into a single implicant, and pairs can sometimes be changed into another pair, or a triple, etc.. Ideally, one would like to combine a pair of implicants into one, since this reduces PLA chip area. However, in a given set of implicants, it is quite possible that no pairs combine. Such a set is said to be a *local* minimum. However, by replacing a pair by another pair, or by a triple, etc., it is possible to create a new set of implicants, where now many combine, leading to a smaller set. The descriptor "local" means there are no implicant sets nearby that contain fewer elements. It is not a "global" minima if there is another set of implicants (somewhere) that has fewer elements.

The derivation of a minimal set is a difficult problem. A heuristic generates a solution to a problem, but not necessarily a minimal one. The PLA minimization problem belongs to the set of NP-complete problems, which means that the best known algorithm for finding the minimal solution increases exponentially in complexity as the problem size increases. It is for this reason that there is so much interest in heuristics. Simulated annealing has the benefit that, under certain conditions, a minimal solution will be found with a probability that approaches 100% as the process continues. This is unlike any other known heuristic for PLA minimization, where, for certain PLA specifications, the probability of finding a minimal solution is 0%.

Figure 3.

PLA Layout Produced by HAMLET.

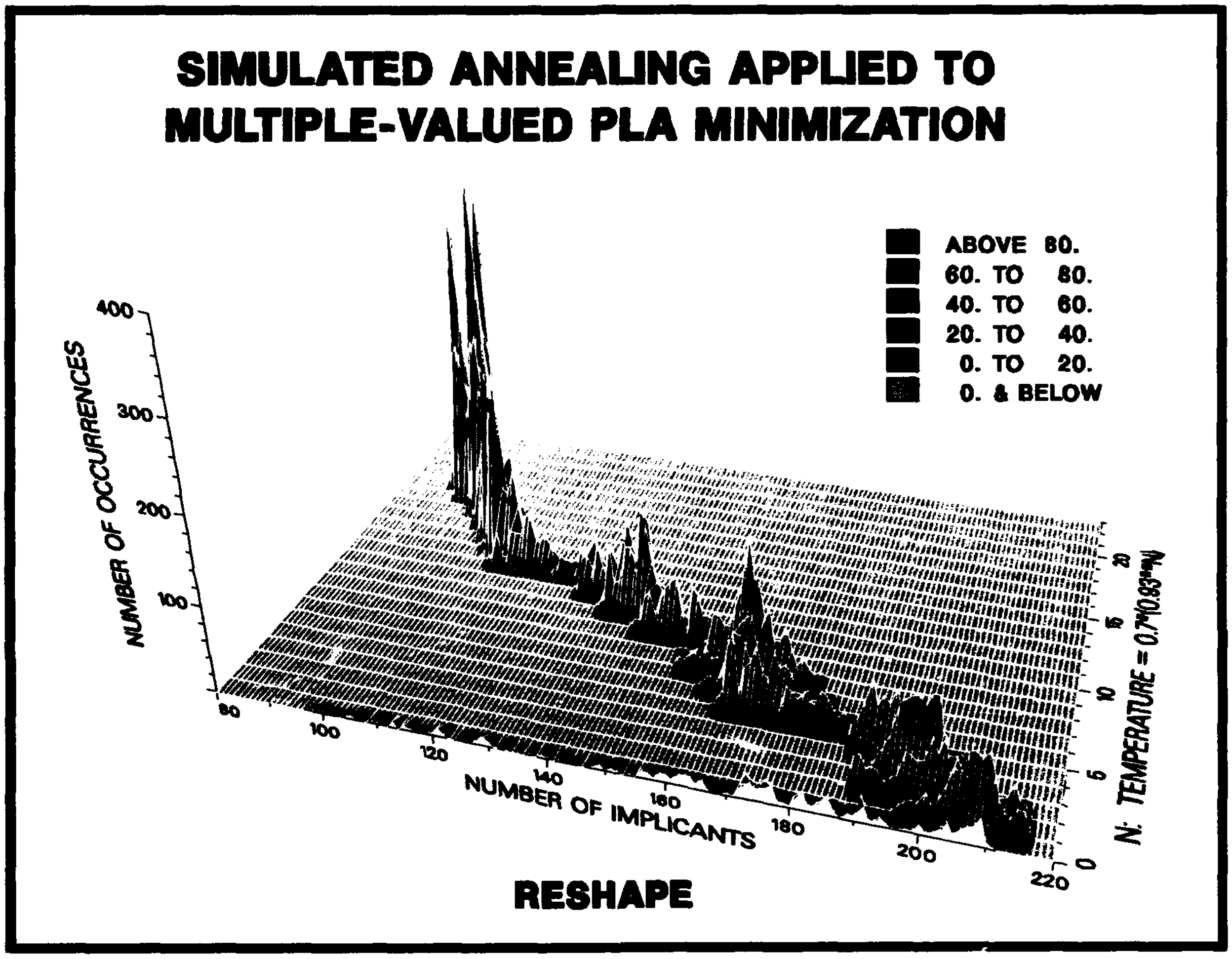


In effect, simulated annealing is a search through a space of solutions. The basic computation is a *move*. That is, given one solution to the PLA specification, another solution is obtained by making a move. Each move begins by first selecting a pair of implicants. If the two implicants can be combined, they are, completing the move. If the implicants cannot be combined, a replacement is considered with the fewest number of equivalent implicants. In effect, a weighted coin is tossed. If the coin lands with heads up, the replacement is made, completing the move. Otherwise, it is not made, and another pair is selected. The analogy with annealing is this. At each temperature, the weighting of the coin is the same for some fixed number of moves. In our programs, we allow 20,000 moves at each temperature. At high temperatures, the weight is such that heads almost always comes up, which means that all moves are accepted. However, at low temperatures, the weight is shifted to tails, so that fewer moves are accepted which increase the number of implicants. At any temperature, a move that reduces the number of implicants is accepted. Thus, at low temperatures, the mixture of generated moves is predominantly PLA complexity-decreasing moves. The existence of complexity-increasing moves at all temperatures is important; it allows the system to escape from local minima. This is analogous to annealing, in which atoms at high temperature move about freely. At lower temperatures the freedom is restricted, as they settle into a low energy state.

Figure 4 shows how simulated annealing performs on a multiple-valued function that begins with 96 implicants. This solution was obtained by applying a heuristic to some given PLA specification. The axis labeled "number of product terms" measures the quality of the solution to the PLA specification; the fewer the better. The axis marked temperature

Figure 4.

Simulated Annealing Applied to Multiple-Valued PLA Minimization.



measures the probability with which implicant increasing moves are accepted. At higher temperatures (toward the front), almost all implicant-increasing moves are accepted. The vertical axis represents the number of times a solution is found at the temperature and the number of implicants on the x-y plane. Here, color encodes the vertical deviation. In effect, the vertical axis shows how the simulated annealing program is doing with respect to the number of product terms. Specifically, at the highest temperature (the slice closest to the front), one can see how the simulated annealing progresses from the initial solution of 96 implicants up to as many as 216 implicants. This is described as *melting*. As the temperature decreases from this point, there is steady progress towards PLA specifications with fewer implicants. In this application of simulated annealing, a solution of 88 implicants is achieved. The behavior of the implicants is similar to the behavior of individual atoms as metal or glass anneals; as the temperature decreases, the form of the final material takes shape as individual atoms occupy a site in the structure corresponding to low energy.

A comparison of simulated annealing with other PLA minimization heuristics shows that it is superior on an average case basis. That is, using HAMLET, random functions were generated, heuristics were applied to all, and the results compared. Simulated annealing surpasses the all known heuristics on the solutions obtained. Although, more time is needed, simulated annealing has the benefit that its performance can be controlled. That is, with larger computation times, one can achieve a more compact realization.

Conclusions

The Navy has a tradition of commitment to computing, as epitomized by the contributions of Rear Adm. Grace Murray Hopper. Indeed, in the past 50 years, significant improvement has occurred in Navy systems because of the computer. It is clear that the next 50 years will bring even further improvement. The promise of compact, high speed computers has inspired our work on multiple-valued logic.

Multiple-valued logic offers a means to overcome significant disadvantages of binary. It makes more efficient use of chip area by encoding more information in the wires that connect devices, and it allows the use of high-speed carry-less operations. Our work at the Naval Postgraduate School has been focused on development of techniques (PLA's) and the tools (CAD) needed to make multiple-valued logic a reality.

Our current effort focuses on expanding simulated annealing. This includes a promising approach to high-speed computations; indeed, initial indications show a modification of the conventional simulated annealing paradigm can produce computation speeds higher than any known heuristic, while maintaining almost the same capability to minimize chip area. We are also pursuing the use of parallel computers to design PLA's[15], including simulated annealing.

Acknowledgements

The author gratefully acknowledges the cooperation of the Naval Research Laboratory in this research. Support for this work has also been provided by the Naval Postgraduate School Direct Funded Research Program, the National Science Foundation, the National Research Council and the North Atlantic Treaty Organization. Special thanks are due the many Naval Postgraduate School students who have contributed to this project, including, LT Hoon-seop Lee, MAJ Yong Ha Ko, LTJG Ugar Ozkan, CAPT Alan W. Watts, LT Yao-Ming Wang, LT S. Onur Oral, LT R. Charles Earle, LT Cem Yilderm, and LT Dave York. Special thanks are also due LCDR John Yurchak, the architect of HAMLET, Professor Chyan Yang, Co-Principal Investigator, and Professor Gerhard Dueck, National Research Council Postdoctoral Associate and collaborator on the simulated annealing research.

Biography

Jon T. Butler is Professor of Electrical and Computer Engineering at the Naval Postgraduate School. He teaches computer design. His research interests include multiple-valued logic and fault tolerant computing. He is a Fellow of the IEEE. He has been Editor of the *IEEE Transactions on Computers* and the *Computer Society Press*, and is the past Editor-in-Chief of *Computer*. Presently, he is the Editor-in-Chief of the *Computer Society Press*. Professor Butler is a co-recipient of the Award of Excellence and the Outstanding Contributed Paper Award for papers presented at the International Symposium on Multiple-Valued Logic in 1985 and 1986, respectively. Currently, he is a member of the Board of Governors of the IEEE Computer Society.

References

1. Besslich, P. W., "Heuristic minimization of MVL functions: A direct cover approach," *IEEE Trans. on Comput.*, February 1986, pp. 134-144.
2. Butler J. T., and H. G. Kerkhoff, "Multiple-valued CCD circuits," *Computer*, Vol. 21, No. 24, pp. 58-69, April 1988.
3. Capasso, F. et al, "Quantum functional devices: resonant tunneling transistors, circuits with reduced complexity, and multiple-valued logic," *IEEE Trans. on Electron. Devices*, Oct. 1989, pp. 2065-2082.
4. Dueck, G. W., R. C. Earle, P. P. Tirumalai, and J. T. Butler, "Multiple-valued programmable logic array minimization by simulated annealing," *Proc. of the 22nd Inter. Symp. on Multiple-Valued Logic*, May 1992, pp. 66-74.
5. Dueck, G. W., and D. M. Miller, "A direct cover MVL minimization using the truncated sum," *Proc. of the 17th Inter. Symp. on Multiple-Valued Logic*, May 1987, pp. 221-227.
6. Kameyama, M., and T. Higuchi, "Prospects of multiple-valued bio-information processing systems," *Proc. of the 18th Inter. Symp. on Multiple-Valued Logic*, May 1988, pp. 360-367.
7. Kameyama, M., T. Sekibe, and T. Higuchi, "Highly parallel arithmetic chip based on multiple-valued bi-directional current-mode logic," *IEEE Jour. of Solid-State Circuits*, Vol. 24, No. 5, Oct. 1989, pp. 1404-1411.
8. Kawahito, S., et al, "A high-speed compact multiplier based on multiple-valued bi-directional current-mode circuits," *Proc. of the 17th Inter. Symp. on Multiple-Valued Logic*, May 1987, pp. 172-180.
9. Kerkhoff, H. G., and J. T. Butler, "Design of a high-radix programmable logic array using profiled peristaltic charge-coupled devices," *Proceedings of the 16th International Symposium on Multiple-Valued Logic*, May 1986, pp. 100-103.
10. Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, No. 4598, 13 May 1983, pp. 671-680.
11. Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *Jour. of Chem. Phys.*, 21, 1953, pp. 1087.
12. Pomper, G., and J. A. Armstrong, "Representation of multivalued functions using the direct cover method," *IEEE Trans. on Comput.* Sept. 1981, pp. 674-679.
13. Tirumalai, P. P., and J. T. Butler, "Minimization algorithms for multiple-valued programmable logic arrays," *IEEE Trans. on Computers*, Feb. 1991, pp. 167-177.
14. Yang, C. and Y.-M. Wang, "A neighborhood decoupling algorithm for truncated sum minimization," *Proceedings of the 20th International Symposium on Multiple-Valued Logic*, May 1990, pp. 153-160, pp. 75-82.
15. Yang, C. and O. Oral, "Experiences of parallel processing with direct cover algorithms form multiple-valued mini-

mization," *Proceedings of the 22nd International Symposium on Multiple-Valued Logic*, May 1992.

16. Yurchak, J. M. and J. T. Butler, "HAMLET – An expression compiler/optimizer for the implementation of heuristics to minimize multiple-valued programmable logic arrays", *Proceedings of the 20th International Symposium on Multiple-Valued Logic*, May 1990, pp. 144-152, For an expanded version, see Yurchak, J. M. and J. T. Butler, "HAMLET user reference manual," Naval Postgraduate School Technical Report NPS-6290-015, July 1990.

17. Zadeh, L. "Fuzzy logic," *Computer*, Vol. 21, No. 24, pp. 83-93, April 1988.

Subsonic and Supersonic Mixing and Combustion Enhancement

Gabriel D. Roy
Office of Naval Research

Abstract

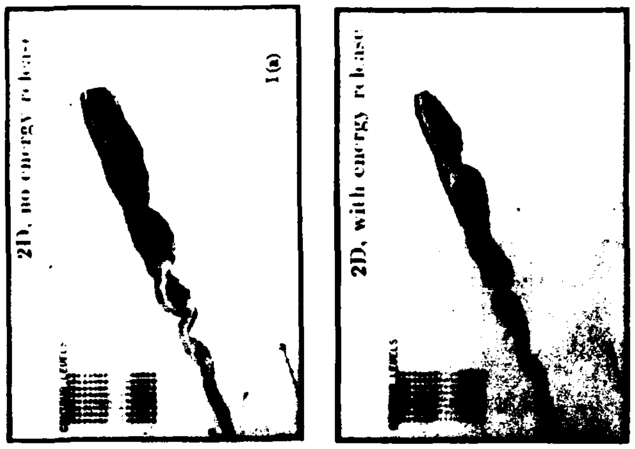
A five year research program was sponsored by the US Department of the Navy to obtain the science base for the conceptual evaluation of higher speed, long range supersonic combustion ramjets. This program was focused to determine the turbulent structure, size scales, intensities and rates of heat and mass transfer of compressible subsonic and supersonic shear layer mixing flows, with and without combustion. Although the program addressed the issues relating to the combustion of solid fuels in supersonic combustion as well, the present paper will focus on supersonic shear layer mixing and combustion enhancement. The studies showed that heat release tends to inhibit the mixing of shear layers. However, three-dimensional excitation at subharmonics of the shear layer, and streamwise vorticity stirring by lobbed splitter plates have been found to increase mixing and combustion. The supersonic shear layers are found to be stable, and their growth deceases, as the convective Mach number (M_c) increases. Vortex generators increase the growth rate of supersonic shear layers by about 30%, whereas shock impingement has very little effect. Non-axisymmetric nozzles and inlets have shown to enhance mixing and combustion of supersonic coaxial flows.

Introduction

As longer ranges are desired for air launched tactical missiles, renewed interest has emerged on air-breathing propulsion systems. Aircraft and ship-imposed constraints on length, diameter and weight of weapons require compact ramjets and scramjets. However, scramjet combustors have been suffering decreased mixing associated with supersonic compressible shear layers, and the associated difficulty in achieving complete combustion in the limited residence times available. To address the fundamental supersonic mixing and combustion processes, the Office of Naval Research (ONR) initiated a focused five year research program. The experimental, analytical and numerical research program addressed the conceptual evaluation of higher speed, longer range supersonic combustion ramjets and the potential for several fold enhancement of the radiant output of high performance aircraft-launched pyrotechnic infrared countermeasures. In particular, the initiative investigated the temporal and spatial scales, and rates of subsonic and supersonic mixing and combustion processes, the supersonic flame structures in mixing layers adjacent to bounding solid surfaces, and the characteristics of the gasification and particle ejection processes from high energy polymeric solid fuels and their boron/metal filled

Figure 1.

Product Concentration without and with Energy Release



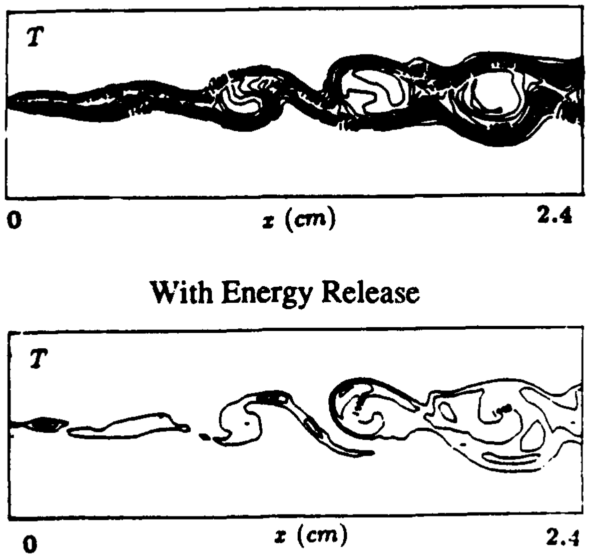
composites into the supersonic mixing layer and flame structure. Among others, the program addressed the following issues, which are specific to this paper:

1. What are the turbulent intensities and size scales in compressible subsonic and supersonic shear layer mixing flows? Are the structures large scale and coherent?
2. Do the streamwise narrow band velocity fluctuations, momentum thickness and mixedness of these turbulent shear layers behave similarly with streamwise distance as in those compressible subsonic shear layer mixing flows?
3. Can the supersonic mixing layers be manipulated by low level forcing at subharmonics of the natural frequency of the mixing layers? Are there other methods of control?
4. Can the compressible subsonic and supersonic mixing layers and flow structures be influenced by combustion generated heat release?

This paper is limited to the research conducted under this special focus program, and complimentary research under the ONR core program. The approach undertaken to solve the issues, and the results obtained by the investigators are summarized. The extensive detailed information obtained from this program is omitted on purpose. The reader may refer to the references given at the end of the paper for further information.

Figure 2.

Temperature Distribution in the Mixing Layer with and without Energy Release

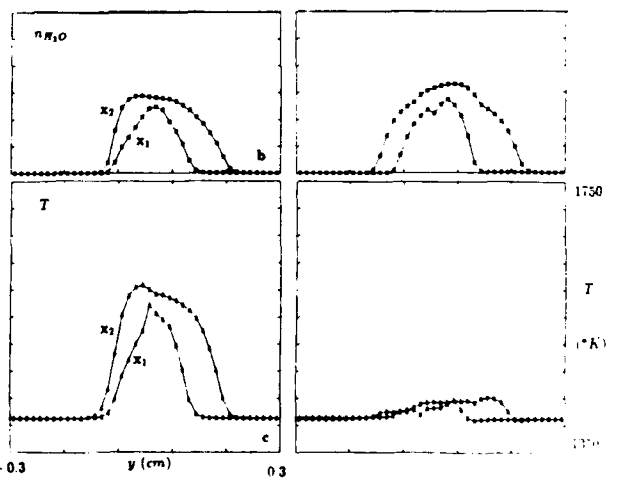


Effect of Heat Release

To understand the dynamics and topology of the coherent structures controlling the development of reactive shear flows, in order to develop both a conceptual model and an analytic framework for their description, Grinstein performed numerical experiments. His approach is to use direct and large-eddy numerical simulation (LES) of spatially evolving three-dimensional compressible shear flows with chemical reaction using monotone 3D flux-corrected transport (FCT) algorithms[1,2]. The details of the Reactive Shear Flow Model and the incorporation of chemistry can be found in Ref. 3. The

Figure 3.

Mean Profiles of Product Concentration and Temperature at Two Streamwise Locations.



time-dependent, compressible, reactive flow conservation equations for total mass and energy density, momentum, and chemical species number densities are solved. The term associated with molecular diffusion, viscous diffusion and thermal conduction in the energy equation are calculated explicitly using central-difference approximations and coupled to convection using timestep-splitting techniques. Since solving a detailed set of chemical kinetic rate equations in conjunction with the unsteady, three-dimensional flow equations is beyond current computer resources, simplified global-chemistry models were used to understand the effects of chemical energy release on the complex three-dimensional flow field.

Reactive Shear Flow Simulations

Hussain and Hussain showed that coherent structures are capable of enhancing or reducing the mixing of coflowing reacting streams, and hence can lead to methods of controlling combustion efficiency through control of the initiation, evolution, and interaction of coherent structures[4]. Chemical reactions between the coflowing streams may also affect the structure of the mixing layer, either through compressibility effects such as baroclinic vorticity production and expansion as energy is released, or by changing the frequency content of the mixing layers with the introduction of characteristic frequencies of the chemical processes. The combustion time scales themselves, in turn, may be modified by the natural acoustic frequencies of the mixing layer or by those imposed through excitation. The interaction between these effects can have dramatic consequences on combustion efficiencies. Earlier studies[5,6] have indicted that the energy released by the chemical reaction has the effect of reducing the entrainment in the mixing layer.

The results of numerical simulations of spatially evolving, chemically reacting mixing layers are presented. The system studied consists of a mixing layer formed by dilute non-premixed subsonic streams of hydrogen and oxygen, which are allowed to react both with and without energy release. Instantaneous results from reactive two-dimensional mixing layer simulations are shown in Fig. 1. Here, the product concentration is plotted against the length of the mixing layer without energy release (Figure 1a), and with energy release (Figure 1b). The reaction considered is $2H_2 + O_2 \rightarrow 2H_2O$, initial temperature = 1400 K, Mach number = 0.3, reactant (molar) concentration is 10%, and the velocity ratio is 3. The flow was organized by forcing the cross stream velocity at the inflow.

Spanwise vortices roll up with the forcing frequency as a result of the nonlinear evolution of the Kelvin-Helmholtz instability, and vortex pairings are inhibited. Thus the growth of the mixing layer is significant only in the vortex-roll-up region. The center of the shear layer gradually shifts towards the upper, slower side as the flow moves downstream, indicating the inherently asymmetric entrainment in the planar shear

Figure 4.

3-D Simulation of Product Generation with Energy Release and Spanwise Excitation.

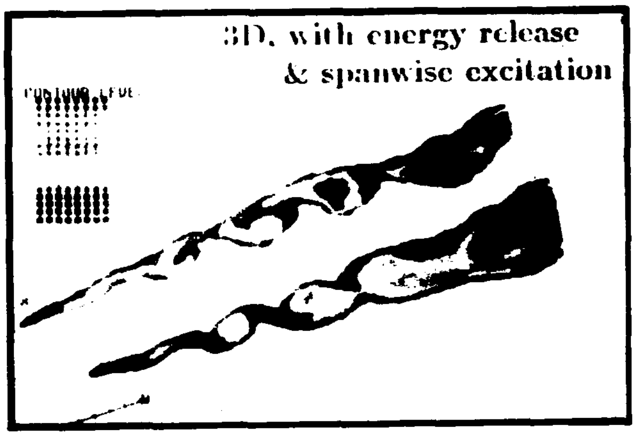
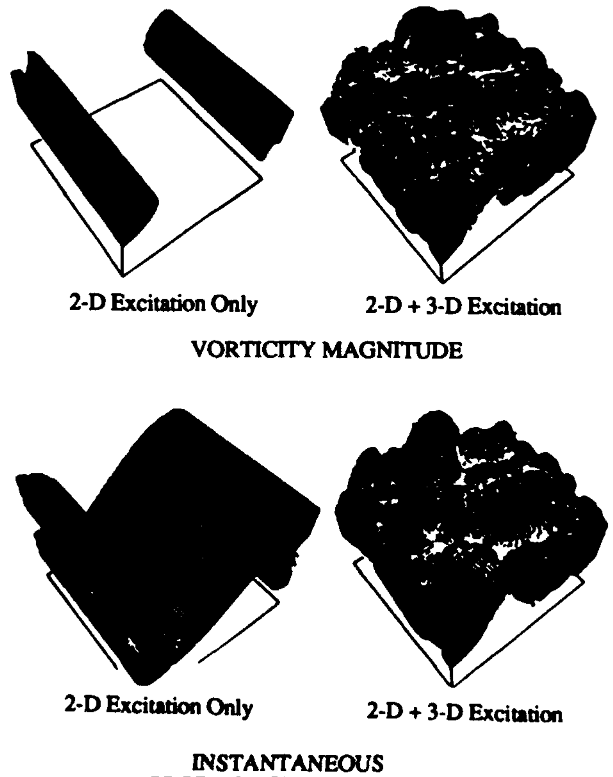


Figure 5.
Enhancement of Vorticity and Chemical Reaction by Secondary Instabilities



flow[7,8]. Due to the fast flow regimes considered, diffusive mixing takes place in the immediate neighborhood of the interface between reactants, and the chemical product generation is observed mainly away from the cores of the structures, especially in its outer edges, which are the most chemically reactive regions. As can be seen from Figure 1, chemical energy release has the effect of reducing the shear layer growth, and the amount of product formed. The product peak-value is found to be 30% larger without energy release. The temperature distribution with and without energy release is shown in Figure 2. It is noted that the chemical energy release is accompanied by significant changes in the temperature, with peak-temperatures at $1.23T_o$ and $1.02T_o$ with and without energy release respectively.

In Figure 3, profiles of time-averaged product concentration and temperature are shown at a streamwise location x_1 and another streamwise location x_2 downstream for cases with and without heat release. These profiles clearly show a shift of the flame towards its slower side, which becomes more pronounced moving downstream. Part of the asymmetry may be attributed to the different mass diffusivities on each side of the shear layer.

However, with spanwise excitation, three-dimensional simulations have shown an increase in mixing even with heat release. For the same conditions used in the previous simulation, the spatially evolving product concentrations are shown for two consecutive time steps in Figure 4. The three-dimensional nature of the mixing layer and mixing enhancement is evident. Spectral simulations were performed by Met Calfe and Hussain to understand the topology and dynamics of large scale coherent structures, and their coupling with fine scale mixing. These simulations have shown that secondary instabilities enhance mixing by convoluting and stretching flame sheet, and they interact with "rolls" to control large scale species transport and reaction rate. Figure 5 shows the vorticity magnitude and instantaneous product generation with 2-D excitation, and with 2-D + 3-D excitation. The three-dimensional excitation produces intense fine scale mixing, as shown by the vorticity magnitude and the instantaneous product generation[9].

Figure 6.

Schematic of the Shear Layer Facility.

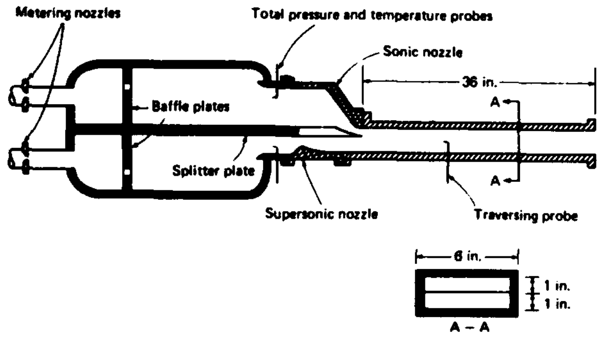


Figure 7.

Velocity Profiles at Various Axial Locations

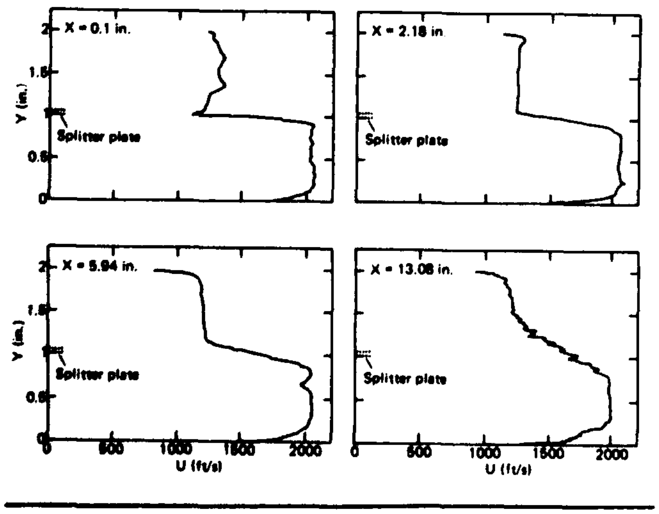
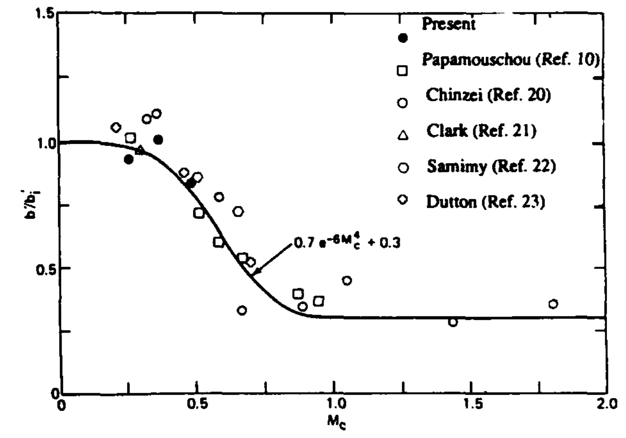


Figure 8.

Experimental Values of Shear Layer Spreading Rate.



Supersonic Mixing Studies

Mixing of Parallel Supersonic Streams

A study was initiated by Waltrup and Sullins to determine the structure and growth of nonreacting shear layers formed when a sonic or supersonic two-dimensional jet mixes with a two-dimensional supersonic air stream. The primary airstream of a typical Mach number of 2 or 3 is separated from a sonic

Figure 9.

Schlieren Photograph of the Shear Layer.

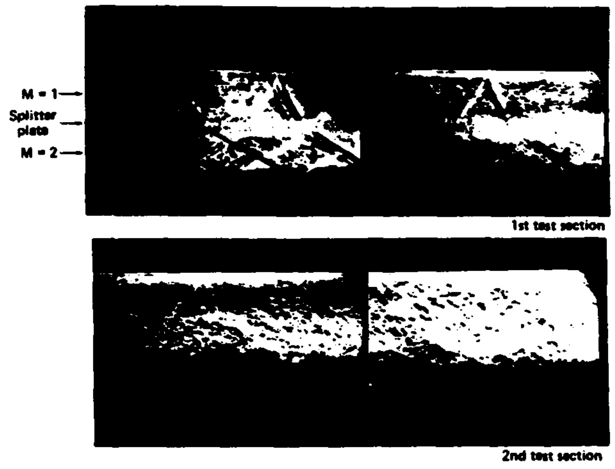
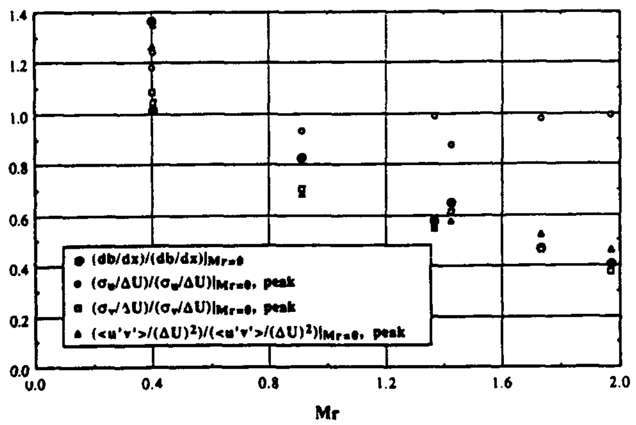


Figure 10.

Normalized Growth Rates and Peak Turbulence Quantities as a Function of Relative Mach Number.



airstream by a sharp trailing edge splitter plate. This is followed by a constant area supersonic mixing section (Figure 6). Diagnostics included Schlieren photography, and pitot and cone static pressure and velocity measurements using a one component laser Doppler velocimeter.

If both streams are incompressible ($M < 0.3$), the shear layer is inherently unstable, and large scale, coherent vortical structures are formed. The growth is dependent on vortex pairing and the ratio of the velocities of the two streams. At supersonic speeds, the shear layer tends to be more stable, and efficient mixing is not achieved. The concept of convective Mach number M_c, established by Roshko and Papamoschou[10] applies.

Tests were conducted with two-dimensional shear layers produced by both Mach 2 and Mach 3 nozzle streams mixing with a Mach 1 nozzle stream. Instream cone static and pitot probes were used to determine the velocity profiles at various axial locations (Figure 7). The velocity profiles were used to determine the shear layer spreading rate (Figure 8). Schlieren photographs were also taken to determine the spreading rate (Figure 9), and the spreading rate determined from the velocity profiles compared well with these photographs.

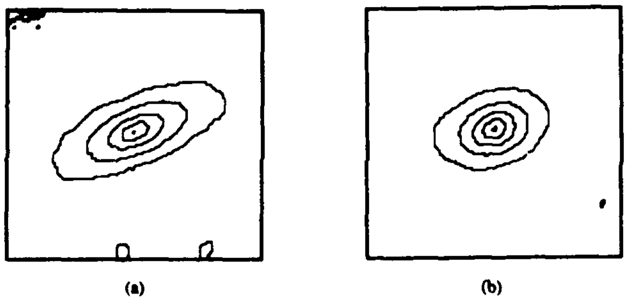
In a series of experiments, Krier and his associates investigated the fundamental fluid dynamic and combustion mechanisms occurring in the entrainment, mixing, and combustion processes in the free shear layer formed between two high speed (supersonic/supersonic or supersonic/subsonic) streams[11].

Mean and Turbulent Velocity Measurements: Velocity measurements have been obtained using a two-component laser Doppler velocimeter system in a two stream, supersonic mixing layer facility. A variety of cases has been examined with free stream velocity ratios in the range from 0.16 to 0.78, freestream density ratios ranging from 0.57 to 1.55, and relative Mach numbers from 0.40 to 1.97. (The relative Mach number is used here as a measure of compressibility effects and is defined as the freestream velocity difference divided by the mean speed of sound, $M_r \equiv \Delta U / \bar{a}$). The range of relative Mach numbers examined spans the range of significant compressibility effects as determined by the reduction in normalized shear layer growth.

Measurements from the developing regions of these shear layers suggest that a local Reynolds number based on freestream velocity difference and mixing layer thickness of $Re_{fd} \equiv \Delta UB/\bar{v} = 1 \times 10^5$ is required for full development of the mean turbulent velocity fields. To determine the growth rates, least-squares linear regressions were performed on the mixing layer thickness data from the fully developed region. The resulting growth rates have been normalized by the growth rate of corresponding incompressible mixing layer at

Figure 11.

2-D Scalar Covariance Field for $M_r = 0.63$. (a) Transverse Image, (b) Oblique Image



the same freestream velocity and density ratios, and these normalized growth rates are plotted in Figure 10 with respect to the relative Mach number. The results clearly show a reduction in the normalized growth rate due to compressibility, which is consistent with those observed by Roshko and others.

Variations in the peak values of the fully developed high-speed mixing layer turbulence quantities from their incompressible counterparts reflect the effects of compressibility on turbulence. The peak values of the streamwise turbulence intensity, transverse turbulence intensity, and the normalized kinematic Reynolds shear stress, $-\langle u' v' \rangle / (\Delta U)^2$, for the compressible cases have been normalized by the respective incompressible values of 0.18, 0.14, and 0.013 and are plotted in Figure 10. From order-of-magnitude estimates using boundary layer approximations to the governing equations, it can be shown that for a compressible mixing layer the normalized kinematic Reynolds shear stress should scale with the normalized growth rate. The comparable reduction in these two quantities with increasing relative Mach number is seen in Figure 10.

Figure 12.

2-D Scalar Covariance Field for M_r = 1.49. (a) Transverse Image, (b) Oblique Image.

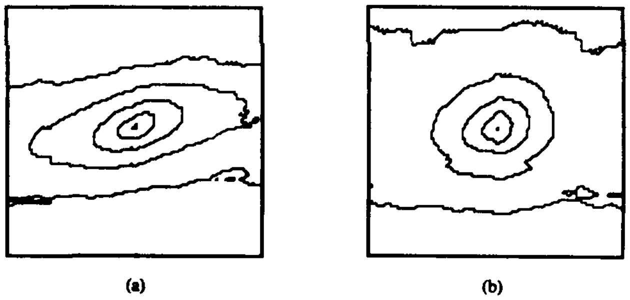
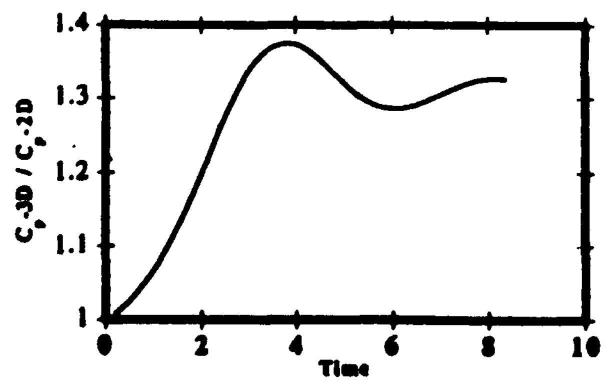


Figure 13.

Comparison of Product Generated with and without 3-D Excitation.

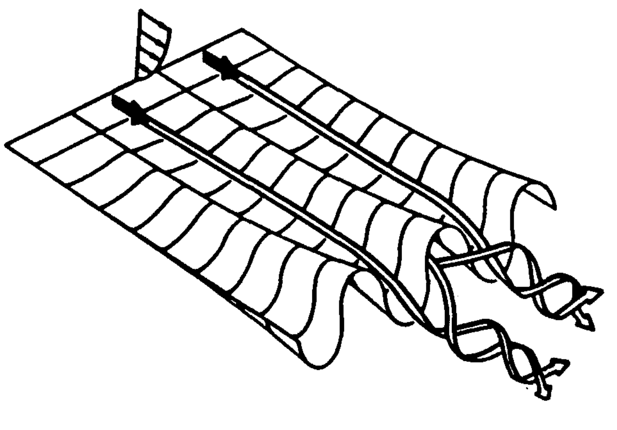


Large Scale Structures: The large scale structures of compressible mixing layers have been studied by Krier's group using planar Mie scattering imaging and statistical analysis of two-dimensional covariance fields[12]. Three flow conditions were investigated, using both traditional transverse imaging as well as oblique imaging. Two types of seeding methods were employed, passive scaler transport of sub-micron ethanol droplets from one freestream to the other as well as the formation of ethanol droplets due to molecular mixing of air from one stream, laden with ethanol vapor, with cold air from the other stream. The relative Mach numbers of the three flow conditions were M_r = 0.63, 1.24 and 1.49, while the local Reynolds numbers for these cases were Re = Δ Ub/ $\bar{v}$ = 3.0 x 10^5 and 6.3 x 10^5 respectively. The level of three-dimensionality in a compressible mixing layer was found to be largely a function of relative Mach number with the Reynolds number apparently having little effect other than to introduce a wider range of turbulent structure scales.

The instantaneous images provide some striking examples of the various flow structures within compressible mixing layers. Organized large scale structures were present in all cases; however they appeared most frequently in the low relative Mach number flow. As the relative Mach number is increased, the transverse images suggest the structures become less organized, with an increased presence of randomly oriented, smaller scale. Structures indicative of streamwise counter-rotating vorticity, and a measure of the level of three-dimensionality of the mixing layer, were more often observed at higher relative Mach number cases. The sizes of these structures also tended to increase with the frequency of occurrence.

Figure 14.

Typical Convoluted Splitter Surface Showing Periodic Streamwise Vortex Array.



The spatial covariance fields of the ensembles of 256 instantaneous images for each case were computed to determine, on a sound statistical basis, the characteristics of the large scale structures, Figure 11 and 12 show the covariance fields for both the transverse and oblique scalar transport images of the low and high relative Mach number cases, respectively. In Figure 11 and 12, the central peak is 100%, and each successive contour is 20% lower. These results indicate that, as the relative Mach number is increased, the transverse scalar transport structures tend to increase in dimensionless size ratio and eccentricity while the angular orientation decreases slightly. These results agree extremely well with recent time-evolving numerical simulations[13] of compressible mixing layers which also show more flattened, elliptical structures at large relative Mach number. Oblique scalar transport correlations indicate structures of generally smaller size ratio and significantly reduced eccentricity than their transversely imaged counterparts, yet the effect of increased relative Mach number indicates only a slight increase in dimensionless structure size with no apparent trend in the eccentricity. The product formation studies exhibited structures, in both the transverse and oblique image planes, which were smaller in dimensionless size and less elliptical in shape than the top stream scalar transport structures. Significantly different characteristics between transverse and oblique realizations of the scalar transport structure implies the presence of an organized, large scale mixing layer structure, which is not entirely due to increased three-dimensionality of the mixing layer associated with increased relative Mach number.

Figure 15.

Comparison of Flame Spreading Achieved with a Flat Splitter and a Convoluted Splitter.

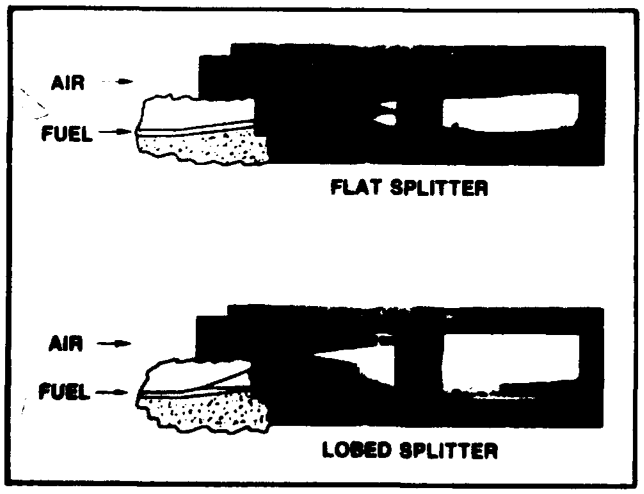


Figure 16.

Vortex Ring Penetration with Transverse Pulse Injection

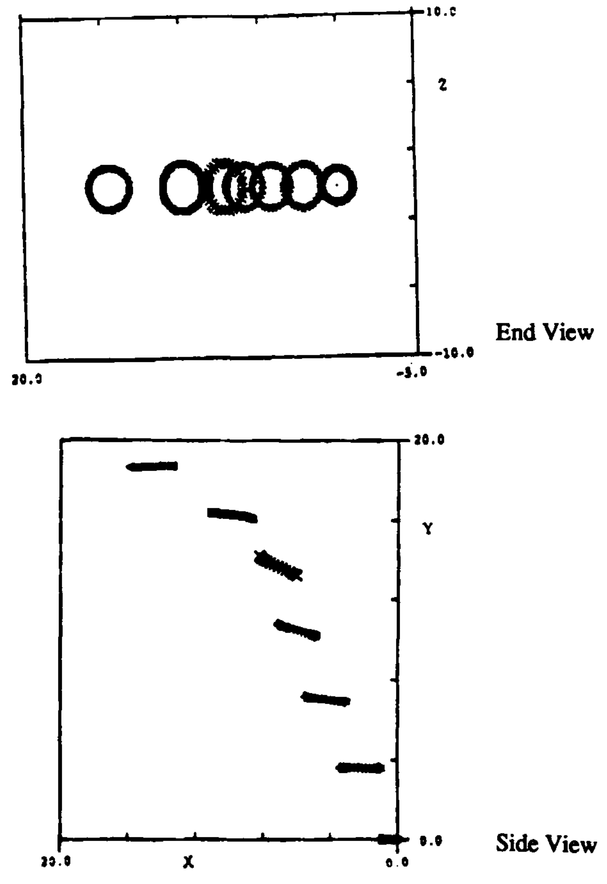
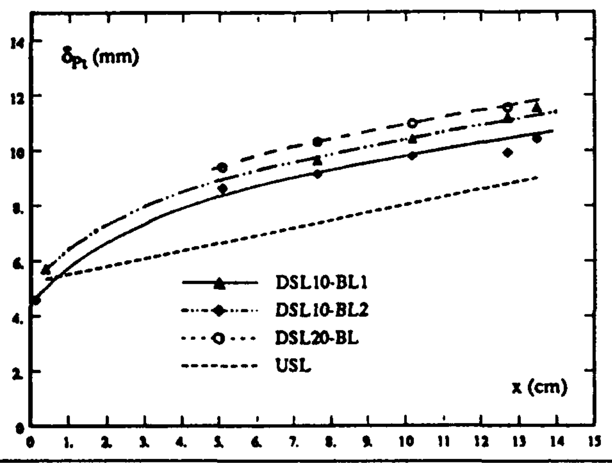


Figure 17.

Pitot Tube Growth with Planar Shock Impingement on boundary Layer.



Mixing and Combustion Enhancement

Subsonic Flows

As mentioned in section II, three-dimensional excitation has been found to increase mixing and combustion. The enhancement in vorticity and product generation by excitation, shown in Figure 5, is further quantified in Figure 13. Here the ratio of product generation with 3-D excitation to that with 2-D excitation is plotted against time. As can be seen, over 30% increase in product generation is obtained by 3-D excitation.

Streamwise Vorticity Stirring: Experimental work by McVey focussed on the effect of streamwise vorticity on the spreading of flames in high speed flows. Using a convoluted splitter surface as shown in Figure 14, and by the proper choice of the amplitude, pitch and profile of the lobed surface, it was possible to generate arrays of very-large-scale, intense vortices. Flow visualizations indicated that the influence of these vortices is first, to cause a large scale exchange of fluid (stirring) followed by a rapid breakdown of the organized structures into random small scale turbulence, which is essential to high speed combustion [14].

Non-premixed flame with air as the oxidant and a mixture of nitrogen and vaporized triethylborane as fuel was used. Since the fuel is pyrophoric, no ignition source was required. Flame photographs with a flat splitter and a lobed splitter are shown in Figure 15. A dramatic increase in the rate of flame spreading is evident, which is attributed to the streamwise vorticity stirring. Further work to understand the details of the mechanisms involved is presently sponsored by the Army Research Office.

Figure18.

Pitot Thickness Growth with 3-D Shock Impingement on Shear Layer.

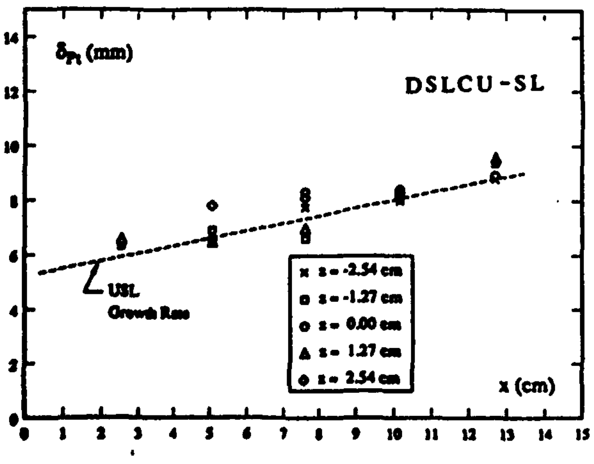


Figure19.

Vortex Generators

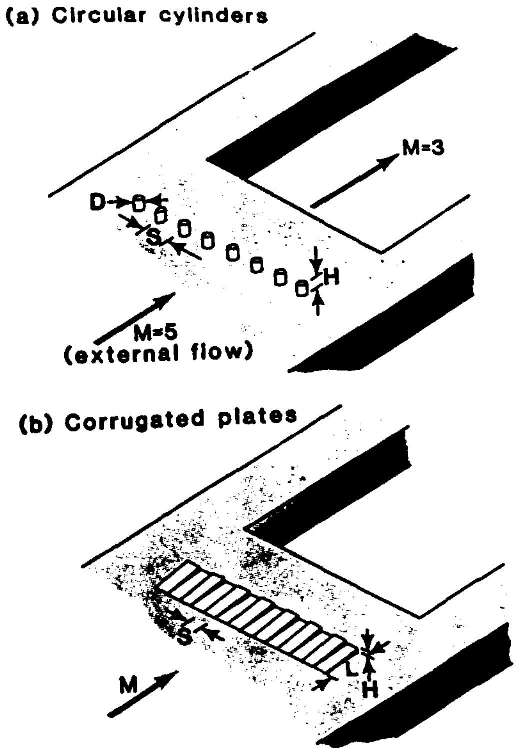
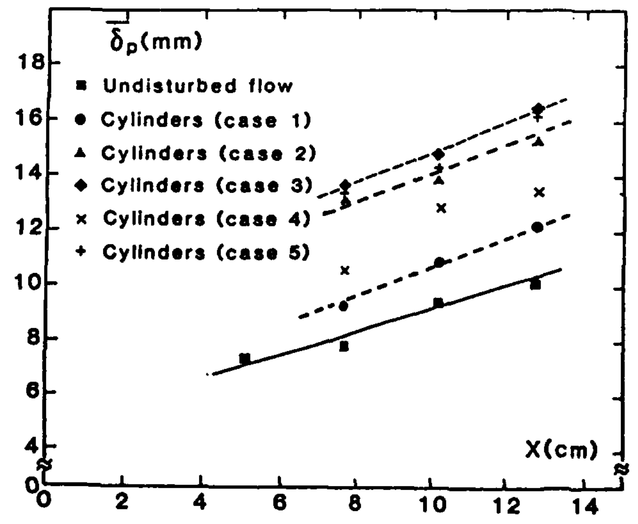


Fig. 19 Vortex Generators

Figure20.

Pitot Thickness Growth with Cylindrical Vortex Generators.



Transverse Pulsed Fuel Injection: Vakili and Wu are studying transverse pulse injection of fuel into the main oxidant stream, as a means of enhancing mixing and combustion. Experiments conducted in subsonic flows indicated that significant penetration of the vortex ring can be obtained by transverse pulse injection (Figure 16). In particular, sharp edge orifices have been found to be very effective. Further studies will be undertaken to determine the effectiveness in super onic mixing enhancement, and consequently supersonic combustion[15].

Supersonic Flows

Compressible shear layers are very stable, and the spreading rate decreases as the convective Mach number increases. In order to achieve complete combustion within the short residence times available in supersonic combustors, it is necessary to enhance mixing. Energy release could further complicate and decrease mixing. Experimental studies have been undertaken by Dolling, and by Schadow to enhance supersonic mixing, and mixing and combustion respectively, by passive or active means.

Dolling's research focused on the growth rate and structure of high Reynold's number turbulent shear layer formed by Mach 5 and Mach 3 airstreams[16]. The first phase of the study was to determine whether the growth rate of the shear layer could be increased by changing the initial conditions at the shear layer origin. The initial conditions were changed either by a planar 3-D shock impingement on the turbulent boundary layer upstream of the origin, or by using cylindrical or wedge shaped vortex generators in the boundary layer upstream of the origin.

Shock Impingement: Both planar and 3-D shock waves were generated and arranged to impinge on either the boundary layer upstream of the shear layer origin or at the shear layer itself. Seven cases were studied. Typical results will be presented here.

For planar shock impingement on boundary layer, three different configurations were used. In Figure 17, the growth rate of the shear layers for these cases are shown. The growth rate of the undisturbed shear layer is also shown, by hatch line, for comparison. A maximum of 30% increase in shear layer growth rate was obtained. However, within a few boundary layer thicknesses downstream, the growth rates are not much different from the undisturbed growth rate.

Three-dimensional shock impingement on the shear layer generated only slight three-dimensionality in the layer immediately downstream of the impingement location due to the pressure gradient caused by the sine wave shape of the incident shock. However, the flow became more two-dimensional further downstream, and the thickness of the shear layer remained about the same. The growth rates across the span are summarized in Figure 18. As can be seen, there are no observable differences between the growth rate and thickness for this case with the undisturbed case.

Vortex Generation: Two types of vortex generators were used. The first consisted of a row of small cylinders screwed into the test surface, and the second consisted of corrugated plates or wedges (Figure 19). The results show that the shear layer growth rate can be increased by about 30% using cylindrical vortex generator arrangements (Figure 20), whereas with wedge-shaped arrangements, the growth rate is unaffected or even reduced.

Non axisymmetric Nozzle Flows: Schadow and his associates have demonstrated the superiority of non axisymmetric nozzles, with respect to circular nozzles, in terms of fine scale and large scale mixing in free-jet and shrouded-jet tests with and without reaction[17,18]. They have extended this work to supersonic jets. The experiments were performed in a coaxial supersonic jet facility, which can be used for nonreacting mixing studies and for supersonic fuel-rich plume combustion studies (Figure 21).

For the mixing studies, the inner (center) jet was kept at $M_1 = 3$. Helium, argon, nitrogen, and heated and unheated air were used to change the density of the inner jet. The velocity

Figure21.

Experimental Set-up (Co-axial jets)

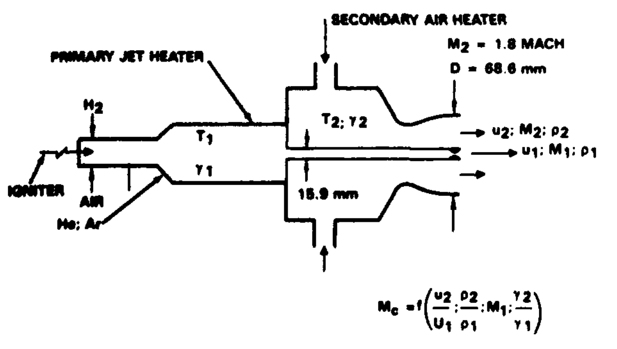


Figure22.

Normalized Spreading Rate of Circular and Noncircular Jets as a Function of Convective Mach Number.

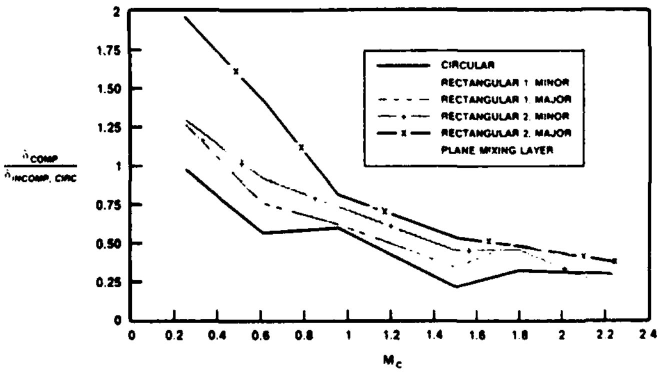


Figure 23.

Radiance Contours of Circular and Multi-step Nozzles.

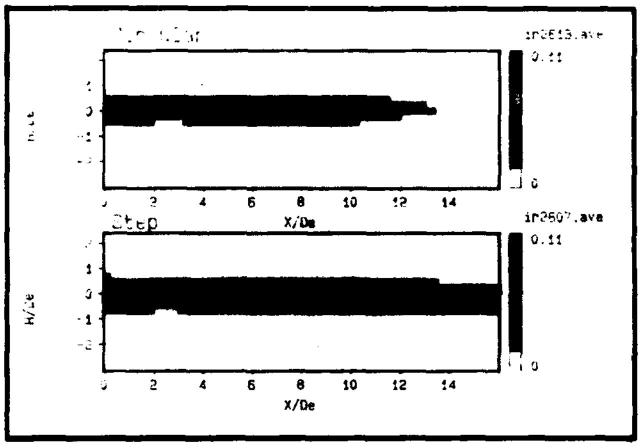
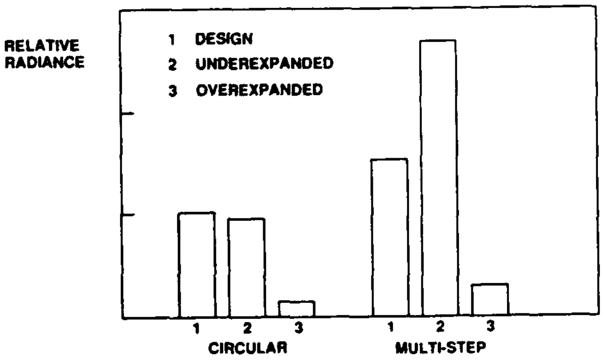


Figure 24.

Relative Total Radiance for Circular and Multi-step Nozzles.



of the outer jet was varied from zero (no coaxial flow) to fully expanded condition (M_2 = 1.8) for ambient and 475K air temperatures. Six test conditions were selected to obtain a convective Mach Number range of 0.5 M_c 2.2. Mixing between the center jet and outer air was determined with a total-pressure probe, and gas sampling probe connected to an analyzer.

In Figure 22, the variation of the normalized spreading rates of the circular jet and the rectangular jets (two configurations) are shown against the convective Mach number. The behavior observed for the circular coaxial jet is similar to that of the planar shear layer, with the normalized spreading rate decreasing with increasing convective Mach number. However, the spreading rates of the rectangular jets are higher than the circular jet, both in the major and minor axis planes, in almost the entire convective Mach number range. The spreading rates are relatively higher for M_c 0.4, because the rectangular jets have a higher incompressible spreading rate than the circular jet.

The supersonic fuel-rich plume combustion experiments were also conducted in the same coaxial jet facility. Here, the outer flow at ambient temperature was fixed at M_c = 1.3 and the inner fuel-rich flow at M_1 = 2. The fuel-rich plume was produced in a gas generator by burning ethylene, air, and oxygen at an equivalence ratio of 2, and a combustion temperature of T_1 = 2300K. Combustion characteristics are compared based on the infrared images, which indicate the spreading rate and combustion intensity. As a typical example, in Figure 23, the radiance map of the underexpanded coaxial jets from a circular and multi-step nozzle are compared [19]. The RMS level of the multi-step nozzle is higher than the circular jet. In Figure 24, the circular and multi-step nozzles are compared for three pressure ratios based on the relative integrated radiance from the nozzle lip to x/D_e = 16. The radiance is normalized by the value for the circular jet at the design pressure ratio. These figures indicate the superior performance of the multi-step nozzle, and the increased heat release associated with enhanced fine-scale mixing.

Summary

A special focus program was sponsored by ONR to understand the mechanisms involved and the control methodology to enhance mixing and combustion is subsonic and supersonic reacting and nonreacting flows. This multi-disciplinary, numerical and experimental research by the academia, Navy and industry laboratories produced the following significant results:

1. Energy release has a tendency to reduce mixing and further product generation.
2. Three-dimensional excitation increases product generation (with energy release).
3. Secondary instabilities are very important in both the core and braid regions of the flow, and they interact with the rolls to control large scale species transport, and thus the reaction rate.
4. Streamwise vorticity stirring by means of convoluted splitter plates has shown a significant improvement in flame spreading.
5. Transverse pulsed injection of the fuel allows deep penetration of the vortex rings into the mainstream, and enhances mixing.
6. Supersonic mixing layers are very stable, and the growth of the mixing layer decreases as the convective Mach number increases.
7. Shock impingement in the boundary layer in supersonic flow produces a small increase in the shear layer growth,

but within a few boundary layer thicknesses downstream, it becomes similar to the undisturbed growth.

8. Supersonic shear layer growth is increased by about 30% by vortex generation, using cylindrical vortex generators.
9. Non axisymmetric nozzles have increased mixing rate as compared to circular nozzles. In coaxial supersonic flows, they have produced increased fine scale mixing, and enhanced combustion.

Further research and development is needed to control and enhance supersonic mixing and combustion. Active, passive, and hybrid control methodologies may be required for specific application. ONR is currently pursuing another special focus program on active control of complex flows.

Acknowledgements

The author acknowledges the valuable contributions made by the various principal investigators, their associates, and graduate students who participated in this program. The excellent cooperation he received from the principal investigators during the management of this program, and the assistance he received from his colleagues Dr. Richard Miller and Dr. Spiro Lekoudis are also acknowledged.

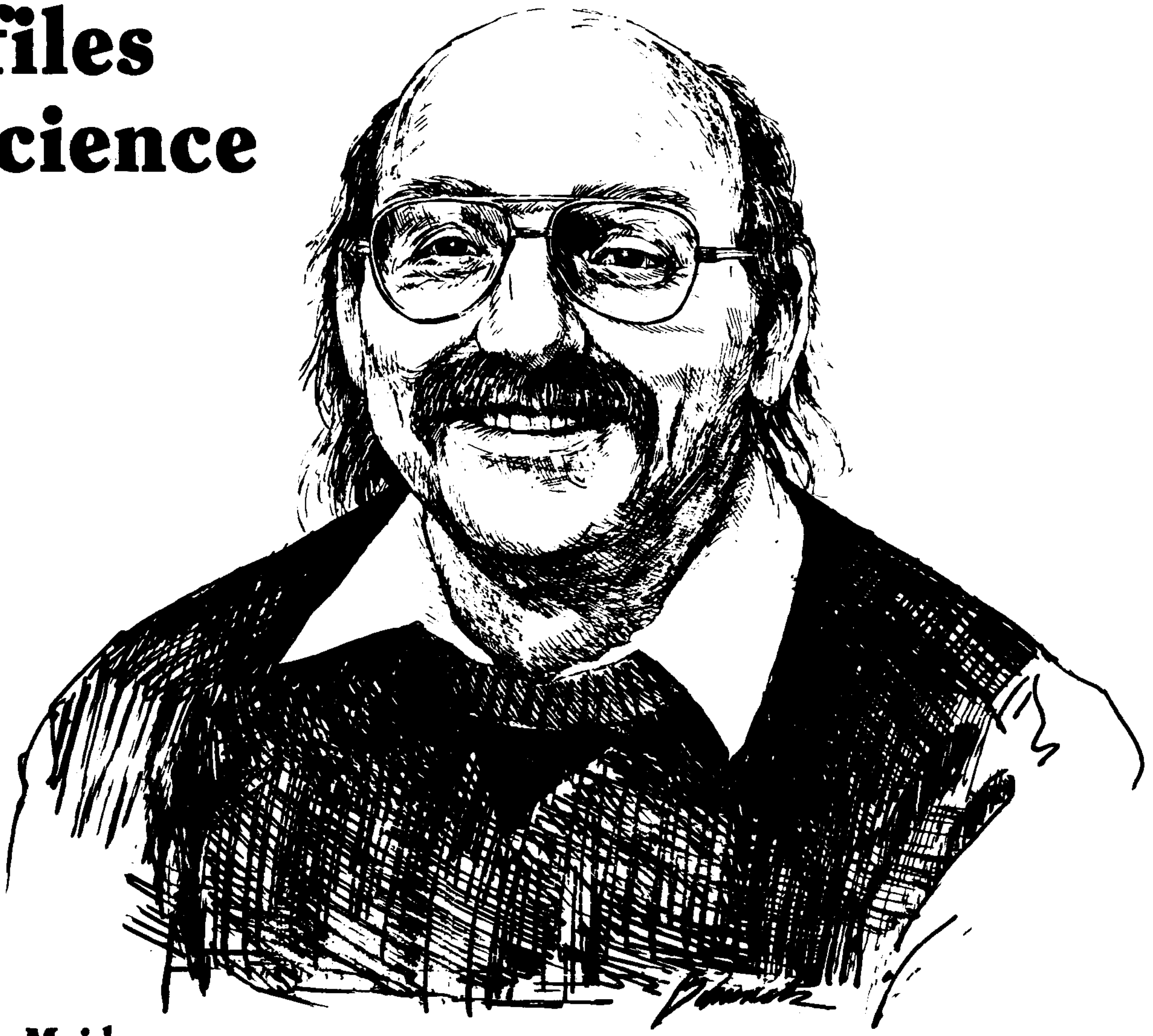
Biography

Gabriel D. Roy received his Ph.D in engineering science from the University of Tennessee. He served as a faculty member for 12 years and a researcher and research manager in university and industry for 15 years. At the Office of Naval Research, Dr. Roy manages the propulsion and pulse power physics programs. He has over 30 publications and two patents, and his research interests include plasma dynamics, MHD, pulse power and propulsion. He is recipient of the ASME Advanced Energy System Jean F. Louis Award and the TRW Roll of Honor Award. Dr. Roy is a member of ASME, and associate fellow of AIAA, and an associate editor of the AIAA *Journal of Propulsion and Power.*

References

1. Boris, J.P., et al., "Direct Simulations of Spatially Evolving Compressible Turbulence." Ninth International Conference on Numerical Methods in Fluid Dynamics, pp. 98-102, Springer Verlag, New York, 1985.
2. Grinstein, F.F., et al., "Three-Dimensional Numerical Simulation of Compressible, Spatially Evolving Shear Flows," Eleventh International Conference on Numerical Methods in Fluid Dynamics pp. 283-287, Springer Verlag, New York, 1989.
3. Grinstein, F. F., et. al., "Numerical Simulation of Transitional and Turbulent Reactive Shear Flows", Proceedings of the Third ONR Propulsion Meeting, Middletown, Rhode Island, 1990.
4. Hussain, A.K.M.F. and H.S. Hussain, "Passive and Active Control of Jet Turbulence," Turbulence Management and Relaminarization, p. 445, Springer Verlag, New York, 1987.
5. McMurtry, P.A. et al., "Direct Numerical Simulations of a Reacting Mixing Layer with Chemical Energy Release," AIAA Journal 24, p. 962, 1986.
6. Hermanson, G.C., M.G. Mungal, and P.E. Dimotakis, "Heat Release Effects on Shear-Layer Growth and Entrainment," AIAA Journal, 25, p. 578, 1986.
7. Grinstein, F.F., E.S. Oran, and J.B. Boris, "Numerical Simulations of Asymmetric Mixing in Planar Shear Flows," Journal of Fluid Mechanics, 25, p. 92, 1986.
8. Koochesfahani, M.M., P.E. Dimotakis, and J.E. Broadwell, "A 'Flip' Experiment in a Chemically Reacting Turbulent Mixing Layer," AIAA Journal, 13, p. 1191, 1985.
9. Hussain, A.K.M.F., Personal Communication, 1990.
10. Papamoschou, D. and A. Roshko, "The Turbulent Compressible Shear Layer" Journal of Fluid Mechanics, 197, p. 453, 1988.
11. Dutton, J.C. and H. Krier, "Supersonic Mixing and Combustion Experiments", Progress Report, 1991.
12. Messersmith, N.L., J.C. Dutton, and H. Krier, "Experimental Investigation of Large Scale Structures in Compressible Mixing Layers," AIAA Paper No. 91-0244, 1991.
13. Burr, R.F. and J.C. Dutton, "Numerical Simulations of Compressible and Reacting Temporal Mixing Layers", AIAA Paper No. 91-1718, 1991.
14. J. McVey, "Accelerated Combustion Through Stream wise Vorticity Stirring", Presentation at ONR, 1988.
15. Vakkili, A. and J.M. Wu, "Scramjet Combustor Mixing Improvements Using Pulsed Transverse-Fuel-Jets Injection", Proceedings of the Third ONR Propulsion Meeting, Middletown, Rhode Island, 1990.
16. Dolling D.S. and Y.R. Shau, "Experimental Study of Growth Rate Enhancement and Structure of Compressible Turbulent Free Shear Layers", Ibid.
17. Gutmark, E., K. C. Schadow, and K.J. Wilson, "Noncircular Jet Dynamics in Supersonic Combustion," AIAA Paper No. 87-1878, 1987.
18. Wilson, K.J., et al., "Mixing Characteristics of Supersonic Shrouded Jets." AIAA Paper No. 88-0699, 1988.
19. Schadow, K.C. Personal Communication, 1991.

Profiles in Science



Andrew J. Majda

Andrew J. Majda is a Professor of Mathematics at Princeton University, where he has taught since 1984. He has been a principal investigator for the Office of Naval Research for many years during which he has done pioneering work on the analytical and computational aspects of turbulence and combustion. He is the recipient of the 1990 John von Neumann Award of the Society for Applied and Industrial Mathematics as well as the 1992 recipient of the National Academy of Science Prize in Applied Mathematics and Numerical Analysis.

Professor Majda is a recognized leader in the world of applied mathematics in his effort to combine rigorous mathematics with state-of-the-art computations. This approach has proved very beneficial and has yielded remarkable elucidation of the theoretical and numerical structure of unstable two-dimensional detonation waves. As a teacher, Professor Majda has passed along his philosophy of applied mathematics to nearly a dozen Ph.D. students who today contribute substantially to the field. See inside front cover, which pertains to his work.

Models for Assessing the Reliability of Computer Software

Nozer D. Singpurwalla and Simon P. Wilson
The George Washington University

Abstract

A formal approach for evaluating the reliability of computer software is through probabilistic models and statistical methods. This paper is an expository overview of the literature on the former. The various probability models for software failure can be classified into two groups; the merits of these groups are discussed and an example of their use in decision problems is given in some detail. The direction of current and future research is contemplated. This research has been supported by the Office of Naval Research, the Army Research Office, and the Air Force Office of Scientific Research.

1. Introduction

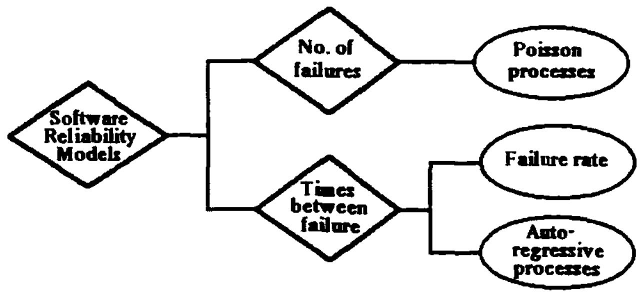
Having been developed over the last 20 years, software reliability is a relatively new area of research for the statistics and the computer science communities. It arose because of interest in trying to predict the reliability of software, particularly when its failure could be catastrophic. Obviously the software that controls an aircraft carrier, a nuclear power station, a submarine or a life-support machine needs to be very reliable, and statistical techniques will aid the computer scientist in deciding if such software has sufficient reliability. The subject is also of commercial importance, as for example when decisions have to be made concerning the release of software into the marketplace.

All software is subject to failure, due to the inevitable presence of errors (or bugs) in the code, so the first aim of the subject has been to develop models that describe software failure. There are various methods of specifying such failure models, and Section 2 discusses these in some detail. It is fair to say that this model derivation has been the focus of research so far. Once a failure model has been specified then it can be applied to problems such as the optimal time to debug software or deciding whether software is ready for release. These applications have received less attention in the literature but are becoming more prevalent. We will mention here that there is another approach to software reliability that differs considerably from the statistical ideas presented here. This approach attempts to prove the reliability, or correctness, of software by formal means of proof, just as one would prove a mathematical theorem. This is an exercise in logic, albeit a rather complex one. It works well on small programs, for example on a program that computes the factorial function, but becomes a forbidding task for even moderately complex pieces of code. Nevertheless, the idea that software can be proved correct is appealing. The approach is not discussed further.

This paper is divided into 5 further sections. Section 2 categorizes the different strategies that have been used to model software failure. Section 3 reviews the historical development of the subject by describing some of the more commonly used models, and Section 4 shows that many of these models can be unified if one adopts a Bayesian position.

Figure 1.

Categorization of Software Reliability Models.



Section 5 looks at applications of the material developed in Section 3, and Section 6 concludes with a look at the current and future direction of the subject. We assume that the reader has some familiarity with some basic reliability and probability concepts; in particular it is important that he or she has knowledge of some common probability distributions, statistical inference and decision making, Poisson processes and the concept of a failure rate.

2. Model Categorization

All statistical software reliability models are probabilistic in nature. They attempt to specify the probability of software failure in some manner. In looking through the literature, one observes that the models developed so far can be broadly classified into two categories.

Type I: Those which propose a probability model for *times between successive failure* of the software, and

Type II: Those which propose a probability model for the *number of failures* up to a certain time. Time is often taken to be CPU time, or the amount of time that the software is actually running, as opposed to real time. In theory, specification via one of these two methods enables one to specify the other. So a model that specifies time between failure will also be able to tell you the number of failures in a given time, and vice versa. In practice, this may not be straightforward.

The first of these categories, modeling time between failure, is most commonly accomplished via a specification of the *failure rate* of the software as it is running. When this is the case the model is to be of Type I-1. The failure rate for the i-th time between failure is given, for i=1, 2, 3,... and a probability model results. One distinctive feature of software is that its failure rate may decrease with time, as more bugs are discovered and corrected. This contrasts with most mechanical systems which will age over time and so have an *increasing* failure rate. An attempt to debug software may introduce more bugs into it, thus tending to increase the failure rate, so the decreasing failure rate assumption is somewhat idealized. However, most of the models of this type that are reviewed here have a decreasing failure rate.

Another way to model time between failure is to define a stochastic relationship between successive failure times. Models that are specified by this method are known as Type I-2, and have the advantage over Type I-1 in that they model the times between failure directly, and not via the abstract concept of a failure rate. For example, let $T_1, T_2, ..., T_i, ...$ be the length of time between successive failure of the software. As a simple case, one could declare that $T_{i+1} = \rho T_i + \varepsilon_i$, where $\rho \geq 0$ is a constant and ε_i is an error term (typically some random variable with mean 0). Then $\rho < 1$ would indicate decreasing time between failure (software reliability expected to become worse), $\rho = 1$ would indicate no expected change in software reliability whilst $\rho > 1$ indicates increasing times between failure (software reliability expected to improve). Those familiar with time series will recognize the relationship in this example as an auto-regressive process of order 1; in general, one would say $T_{i+1} = f(T_1, T_2, ..., T_i) + \varepsilon_i$ for some function f.

The second of these categories, modeling the number of failures, uses a point process to count the failures. Let M(t) be the number of failures in the software that are observed during time [O,t). M(t) is modeled by a *Poisson process*, which is a stochastic process with the following properties.

i) $M(0) = 0$ and if $s < t$ then $M(s) \leq M(t)$. M(t) takes values in $\{0, 1,2,...\}$

ii) The number of failures that occur in disjoint time intervals are independent. So, for example, the number of failures in the first 5 hours of use has no effect on the number of failures in the next 5 hours.

iii) The number of failures to time t is a Poisson random variable with mean $\mu(t)$, for some non-decreasing function $\mu(t)$; that is to say:

$$P(M(t) = n) = \frac{(\mu(t))^n}{n!} e^{-\mu(t)} \qquad n = 0,1,2,\ldots$$

The different models of this type have a different function $\mu(t)$, which is called the mean value function. The mean number of failures at time t is indeed $\mu(t)$, as is the variance. The Poisson process is chosen because in many ways it is the simplest point process yet it is flexible and has many useful properties that can be exploited. This second approach has become increasingly popular in recent years. M(t) can also be specified by its intensity function $\lambda(t)$, which is the derivative of $\mu(t)$ with respect to t; either of these functions completely specify a particular Poisson process. One disadvantage of this approach is that it implies that there are conceptually an infinite number of bugs in the program, which is obviously impossible for code of a finite length. Another disadvantage is more subtle; the model implies a positive correlation between the number of failures in adjoining time intervals, a situation which is not true since again the total number of bugs has to be finite.

Figure 1 is a flow-chart showing the above categorization of the statistical models.

Figure 2(i).

The failure rate of the model of Jelinski and Moranda

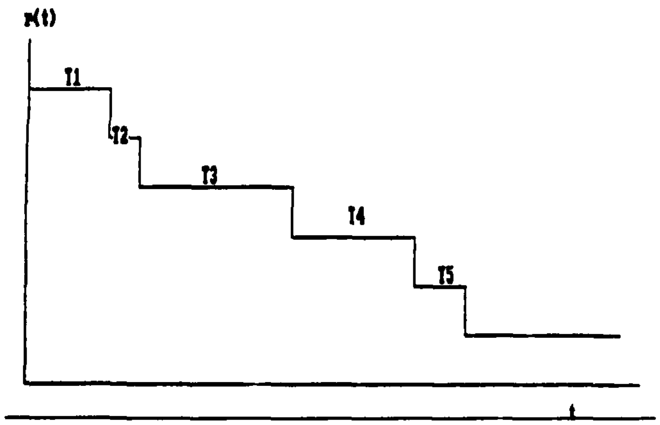
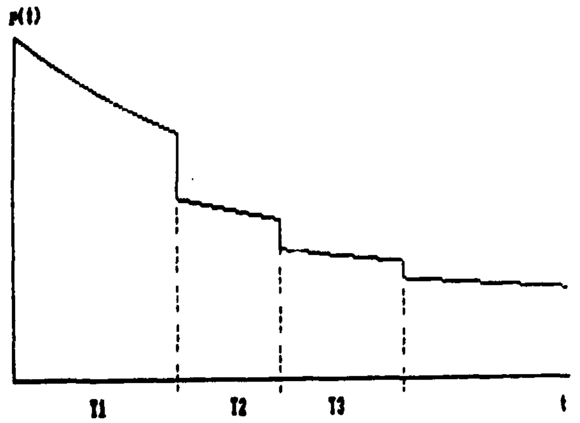


Figure 2(ii).

The failure rate of the model of Littlewood and Verall.



3. Review of Some Software Reliability Models

This section introduces some of the more well known probability models for software reliability. There are examples of models from each of the two main categories that were discussed in the previous section. Since the main purpose of the review is to describe the ideas and assumptions behind the models, technical details will be kept to a minimum in most cases. Those interested in the details of a particular model are advised to reference the papers where they were originally presented.

Some common notation will be assumed throughout this section and is given below:

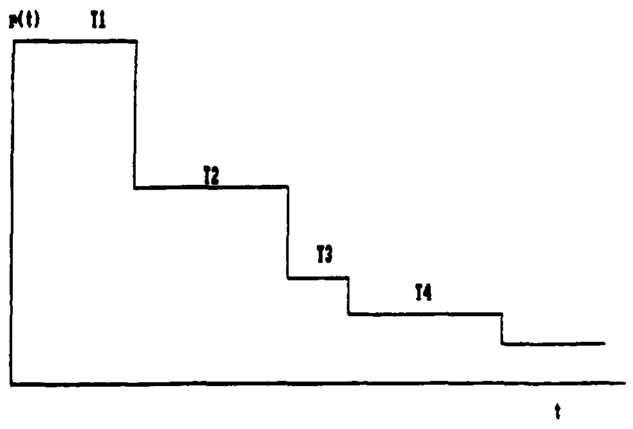
i) T_i= i-th time between failure of the software [i.e. time between (i-1)th and i-th failure].

ii) $r_{T_i}(t)$= failure rate for T_i, the i-th time between failure, at time t.

iii) M(t)= number of failures of the software in the time interval [0, t) (a Poisson process).

Figure 2(iii)

The failure rate of the model of Moranda.



iv) $\lambda(t)$= intensity function of M(t).

v) $\mu(t)$= expected number of failures of software in time [0,t).

$$= \int_0^t \lambda(s)\, ds, \textit{ since } M(t) \textit{ is a Poisson process.}$$

10 models are presented. Model numbers 1 to 7 are of Type I-1, models 8 and 9 are of Type II and model 10 is of type I-2. A common problem to all the models is the lack of data on which to test their validity; data on software reliability is commercially sensitive and so statisticians in academia have very little information on how software in the marketplace actually performs. For this reason it is important that the assumptions made in deriving these models are carefully thought about.

The Model of Jelinski & Moranda (1972).

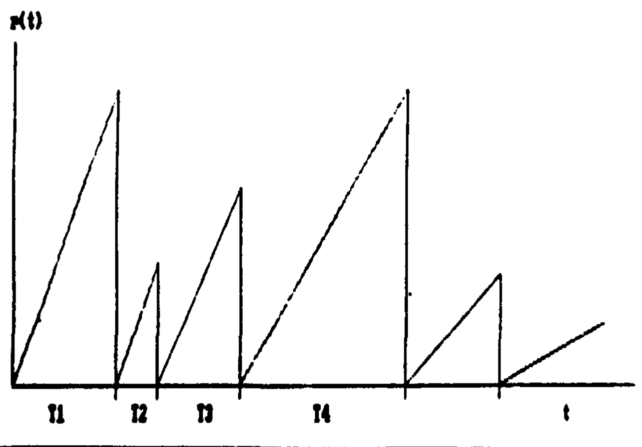
This was the very first software reliability model that was proposed, and has formed the basis for many models developed after. It is a Type I-1 model; it models times between failure by considering their failure rates. Jelinski and Moranda reasoned as follows. Suppose that the total number of bugs in the program is N, and suppose that each time the software fails, one bug is corrected. The failure rate of the i-th time between failure, T_i, is then assumed a constant proportional to N-i+1, which is the number of bugs remaining in the program. In other words

$$r_{T_i}(t \mid N, \Lambda) = \Lambda\,(N - i + 1), \quad i = 1,2,3,\ldots, t \geq 0,$$

for some constant Λ.

Figure 2(iv).

The failure rate of the model of Schick and Wolverton.



There are some criticisms that one could make of the model. It assumes that each error contributes the same amount to the failure rate, whereas in reality different bugs will have different effects. It also assumes that every time a fix is made, no new bugs are introduced; note [see Figure 2(i)] that the successive failure rates are indeed decreasing. A model like this is sometimes referred to as a "de-eutrophication model", because the process of removing bugs from software is akin to the removal of pollutants in rivers and lakes.

Bayesian Reliability Growth Model (Littlewood & Verall (1973)).

Like the Jelinski & Moranda model, the model proposed by Littlewood and Verall looked at times between failure of the software. However, they did not develop the model by

Figure 2(v).

The intensity function for the model of Goel and Okumoto.

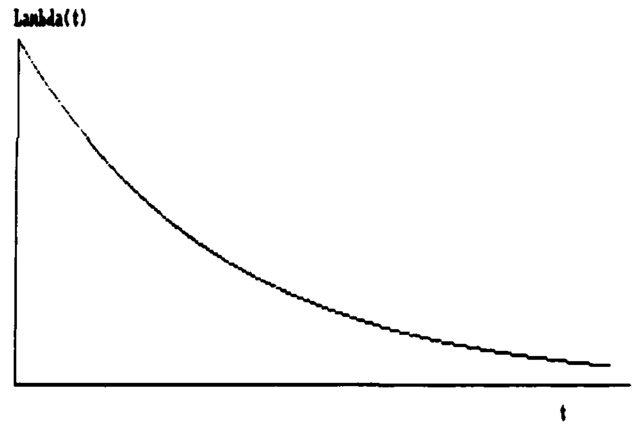
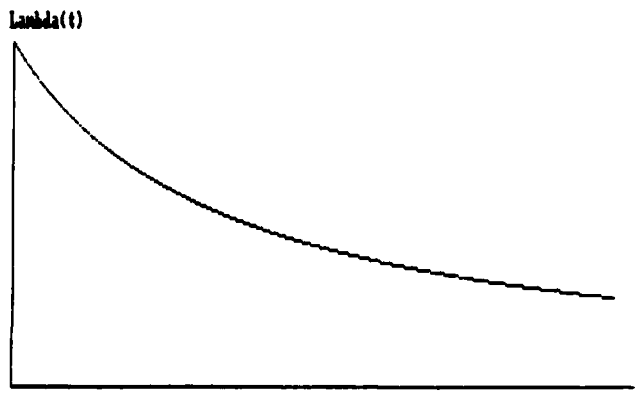


Figure 2(vi).

The intensity function for the model of Musa and Okumoto.



characterizing the failure rate; rather they stated that the model should *not* be based on fault content (as Jelinski & Moranda had assumed) and then declared that T_i has an exponential distribution with scale Λ_i, and that Λ_i itself has a gamma distribution with shape α and scale $\psi(i)$, for some function ψ. Despite this it is still considered to be a Type I-1 model.

Specifically:

$$f_{T_i}(t \mid \Lambda_i) = \Lambda_i e^{-\Lambda_i t} \quad t \geq 0$$

$$\Pi_{\Lambda_i}(\lambda \mid \alpha, \psi(i)) = \frac{(\psi(i))^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\psi(i)\lambda} \quad \lambda \geq 0$$

$\psi(i)$ was supposed to describe the quality of the programmer and the programming task. As an example, they chose $\psi(i) = \beta_0 + \beta_1 i$. One can show that this makes the failure rate of each T_i decreasing in t and that each time a bug is discovered and fixed there is a downward jump in the successive failure rates; see Figure 2(ii). In fact

$$r_{T_i}(t \mid \alpha, \beta_0, \beta_1) = \frac{\alpha}{\beta_0 + \beta_1 i + t}, \text{ for } t \geq 0.$$

If $\beta_1 > 1$ then the jumps in the failure rate decrease in i, if $\beta_1 < 1$ they increase whilst if $\beta_1 = 1$ they remain a constant. So if β_1 differs from 1 then the fixing of each bug is making a different contribution to the reduction in the failure rate of the software, an apparent advantage over the model by Jelinski & Moranda. This model has received quite a lot of attention and has been the subject of various modifications; see models 6 and 7 later in this section.

The De-Eutrophication model of Moranda (1875).

Another (de-eutrophication) model of Moranda (1975) attempted to answer some of the criticisms of the Jelinski & Moranda model, in particular the criticism concerning the equal effect that each bug in the code has on the failure rate. He hypothesized that the fixing of bugs that cause early failures in the system reduces the failure rate more than the fixing of bugs that occur later, because these early bugs are more likely to be the bigger ones. With this in mind, he proposed that the failure rate should remain constant for each T_i, but that it should be made to decrease geometrically in i after each failure i.e. for constants D and k.

$$r_{T_i}(t \mid D, k) = Dk^{i-1} \quad t \geq 0, D > \theta \text{ and } 0 < k < 1.$$

Compared to the Jelinski & Moranda model, where the drop in failure rate after each figure was always Λ, the drop in failure rate here after the i-th failure is D k^{i-1}(1-k) see Figure 2(iii). The assumption of a perfect fix, with no introduction of new bugs during the fix, is retained.

Imperfect Debugging Model (Goel & Okumoto (1978)).

This model is another generalization of the Jelinski & Moranda model which attempts to address the criticism that a perfect fix of a bug does not always occur. Goel & Okumoto's *Imperfect Debugging Model* is like the Jelinski & Moranda model, but assumes that there is a probability p, $0 \leq p \leq 1$, of fixing a bug when it is encountered. This means that after i faults have been found, we expect i x p faults to have been corrected, instead of i. Thus the failure rate of T_i is

$$r_{T_i}(t \mid N, \Lambda, p) = \Lambda(N - p(i-1))$$

When p =1 we get the Jelinski & Moranda model.

A model by Schick & Wolverton (1978).

This is yet another Type I model, and this time the failure rate is assumed proportional to the number of bugs remaining in the system and the time elapsed since the last failure. Thus

$$r_{T_i}(t \mid N, \Lambda) = \Lambda(N - i + 1)t, \qquad t \geq 0$$

This model differs from models 1-4 in that the failure rate does not decrease monotonically. Immediately after the i-th failure, the failure rate drops to 0, and then increases linearly with slope (N-i) until the (i+1)th failure; see Figure 2(iv).

Bayesian Differential Debugging Model (Littlewood (1980)).

This model can be considered as an elaboration of model 2 proposed by Littlewood & Verall. Recall that in model 2 it was assumed that Λ_i, the failure rate of the i-th time between failures, was declared to have a gamma distribution. In this new model Littlewood supposed that there were N bugs in the system (a return to the bug counting phenomenon), and then proposed that Λ_i be specified as a function of the remaining bugs. In particular, he stated $\Lambda_i = \phi_1 + \phi_2 + ... + \phi_{N-i}$, where ϕ_i were independent and identically distributed gamma random variables with shape α and scale β. This implied that Λ_i would have a gamma distribution with shape $\alpha(N-i)$ and scale β. In other respects its assumptions are identical to the original Littlewood/Verall model.

Bayes Empirical Bayes or Hierarchical Model (Mazzuchi & Soyer (1988)).

In 1988 Mazzuchi & Soyer proposed a *Bayes Empirical Bayes or Hierarchical* extension to the Littlewood & Verall model. As with the original model, they assumed T_i to be exponentially distributed with scale Λ_i. Then they proposed two ideas for describing Λ_i, here called model A and model B.

Model A

Still assume that Λ_i is described by a gamma distribution, but with parameters α and β. Now assume that α and β are independent and that they themselves are described by probability distributions; α by a uniform and β by another gamma. In other words:

$$\Pi_{\Lambda_i}(\lambda \mid \alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}, \quad \lambda \geq 0$$

$$\pi(\alpha \mid v) = \frac{1}{v}, \quad 0 \leq \alpha \leq v$$

$$\pi(\beta \mid \alpha, b) = \frac{b^{a}}{\Gamma(a)} \beta^{a-1} e^{-b\beta}, \quad \beta \geq 0, a > 0, b > 0.$$

Model B

Assume that Λ_i is described exactly as in Littlewood and Verall i.e.

$$\Pi_{\Lambda_i}(\lambda \mid \alpha, \psi(i)) = \frac{(\psi(i))^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\psi(i)\lambda}, \quad \lambda \geq 0$$

and that $\psi(i) = \beta_0 + \beta_1 i$, except now place probability distributions on α, β_0 and β_1 as follows:

$$\pi(\alpha \mid \omega) = \frac{1}{\omega}, \quad 0 \leq \alpha \leq \omega$$

$$\pi(\beta_0 \mid a,b,\beta_1) = \frac{b^{a}}{\Gamma(a)} (\beta_0 + \beta_1)^{a-1} e^{-b(\beta_0+\beta_1)},$$

$$\beta_0 \geq -\beta_1, a > 0, b > 0$$

$$\pi(\beta_1 \mid c,d) = \frac{d^{c}}{\Gamma(c)} \beta_1^{\,c-1} e^{-d\beta_1}$$

$$\beta_1 \geq 0, c > 0, d > 0.$$

So α is described by a uniform distribution, β_0 by a shifted gamma and β_1 by another gamma, and there is dependence between β_0 and β_1. By assuming β_1 to be degenerate at 0, model A is obtained from model B. The authors were able to find an approximation to the expectation of T_{n+i} given that $T_1=t_1, T_2=t_2, \ldots, T_n=t_n$, and so use their model to predict future reliability of the software in light of the previous failure times.

Time-dependent Error Detection Model (Goel & Okumoto (1979)).

This is the first Type II model that we will consider. It assumes that M(t), the number of failures of the software in time [0,t), is described by a Poisson process with intensity function given by

$$\lambda(t) = ab\, e^{-bt}$$

where a is the total expected number of bugs in the system and b is the fault detection rate; see Figure 2(v). Thus the expected number of failures to time t is:

$$\mu(t) = \int_0^t ab\, e^{-bs}\, ds = a(1 - e^{-bt}).$$

The function $\mu(t)$ completely specifies a particular Poisson process, and the distribution of M(t) is given by the well known formula

$$P(M(t) = n) = \frac{(\mu(t))^n}{n!} e^{-\mu(t)}, \qquad n = 0,1,2,\ldots$$

Experience has shown that often the rate of faults in software increases initially before eventually decreasing, and so in Goel (1983) the model was modified to account for this by letting

$$\lambda(t) = abc\, t^{c-1} e^{-bt^{c}}$$

where a is still the total number of bugs and b and c describe the quality of testing.

Logarithmic Poisson Execution Time Model (Musa and Okumoto (1984)).

The *Logarithmic Poisson Execution Time Model* of Musa and Okumoto is one of the more popular software failure models of recent years. It is a type II model, but the model is not derived by directly assuming some intensity function $\lambda(t)$, as was the case with the model of Goel & Okumuto. Here $\lambda(t)$ is expressed in terms of $\mu(t)$, the expected number of failures in time [0,t) via the relationship.

$$\lambda(t) = \lambda_0 e^{-\theta\mu(t)}.$$

Put simply, this relationship encapsulates the belief that the intensity (or rate) of failures at time t decreases exponentially with the number of failures experienced, and so bugs fixed, up to time t. The fixing of earlier failures will reduce $\lambda(t)$ more than the fixing of later ones. Since we are modeling the number of failures by a Poisson process, then we have another relationship between $\lambda(t)$ and $\mu(t)$, namely

$$\mu(t) = \int_0^t \lambda(s)\, ds.$$

Using these two relationships between $\lambda(t)$ and $\mu(t)$, there is a unique solution for the two functions:

$$\lambda(t) = \frac{\lambda_0}{\lambda_0 \theta t + 1} \quad ; \quad \mu(t) = \frac{1}{\theta} \ln(\lambda_0 \theta t + 1).$$

Figure 2(vi) shows a plot of $\lambda(t)$ versus t; it is similar to the plot of figure 2(v) except that the tail is thicker.

It now follows from the above that by using $P(M(t) = n) = (\mu(t))^n e^{-\mu(t)}/n!$ we can say

$$P(M(t) = n) = \frac{(\ln(\lambda_0 \theta t + 1))^n}{\theta^n (\lambda_0 \theta t + 1)^{1/\theta} x\, n!}, \quad n = 0, 1, 2 \ldots$$

As a final remark, we mention that in their paper the authors go into some detail on estimation of λ_0 and θ by maximum likelihood methods; however, one of the likelihoods appears to be incorrect.

Random Coefficient Autoregressive Process Model (Singpurwalla & Soyer (1985)).

This is a Type I-2 model, that is one that does not consider the failure rate of times between failure. Instead it assumes that there is some pattern between successive failure times and that this pattern can be described by a functional relationship between them. The authors declare this relationship to be of the form

$$T_i = T_{i-1}^{\theta_i} \quad , \quad i = 1, 2, 3, \ldots$$

where T_0 is the time to the first failure and θ_i is some unknown coefficient. If all the θ_i's are bigger than 1 then we expect successive lifelengths to increase, and if all the θ_i's are smaller than 1 we expect successive lifelengths to decrease.

Uncertainty in the above relationship is expressed via an error term δ_i, so that

$$T_i = \delta_i T_{i-1}^{\theta_i}.$$

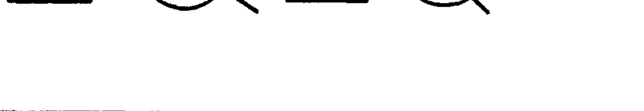
The authors then make the following assumptions, which greatly facilitate the analysis of this model. They assume the T_i's to be lognormally distributed, that is to say that log T_i's have a normal distribution, and that they are all scaled so that $T_i \geq 1$. The δ_i's are also assumed to be lognormal, with median 1 and variance σ_1^2 (the conventional notation is $\Lambda(1, \sigma_1^2)$). Then by taking logs on the relationship above they obtain

$$\log T_i = \theta_i \log T_{i-1} + \log \delta_i$$

$$= \theta_i \log T_{i-1} + \varepsilon_i \quad \text{, say.}$$

Figure 3.

Decision process for single-stage testing.



Since the T_i's and the δ_i's are lognormal so the log T_i's and the ε_i's (=log δ_i's) will be normally distributed, and in particular ε_i has mean 0 and some variance σ^2 (the conventional notation is $N(0, \sigma^2)$). The log-lifelengths therefore form what is known as an *autoregressive process of order 1 with random coefficients* θ_i. There is an extensive literature on such processes which can now be used on this model.

All that remains to do is to specify θ_i, and the authors consider several alternative models. For example, one could make θ_i itself an autoregressive process:

$$\theta_i = \alpha\theta_{i-1} + \omega_i \quad , \quad \text{where } \omega_i \textit{ is } N(O, W_i) \text{ with } W_i \text{ known.}$$

When α is known, the expressions for log T_i, and θ_i together form a *Kalman filter model*, on which there is also an extensive literature. When α is not known the solution is via an *adaptive Kalman filter* algorithm for which the authors propose an approach. As an alternative to the above, one could place a two stage distribution on θ_i, and the authors considered the idea of θ_i being $N(\lambda, \sigma_2^2)$, with λ also a normal random variable having mean m_0 and variance s_0^2. In this latter case one can employ standard hierarchical Bayesian inference techniques to predict future reliability in the light of previous failure data.

4. Model Unification.

By adopting a Bayesian approach, it turns out that one can unify models 1, 2 and 8 - the models by Jelinski & Moranda, Littlewood & Verall and Goel & Okumoto respectively - under a general framework. Observe that this also provides a link between the two types of models, since models 1 and 2 are of type I whilst model 8 is of type II.

We begin by recalling the first model, that by Jelinski & Moranda. Each T_i is assumed to have a constant failure rate $\Lambda(N-i+1)$. It is well known that this implies each T_i must therefore be exponentially distributed with mean $(\Lambda(N-i+1))^{-1}$. Now assume that Λ and/or N is unknown; in true Bayesian fashion prior distributions are placed upon them.

To obtain model 8 by Goel & Okomuto, we let Λ be degenerate at λ and N have a Poisson distribution with mean θ. One can calculate M(t) using the T_i's as defined by Jelinski & Moranda, and then by averaging out over N one finds that M(t) is indeed a Poisson process with mean:

$$\mu(t) = \theta(1 - e^{-\lambda t})$$

which is the form of $\mu(t)$ for Goel & Okumoto's model.

One can also obtain model 2 by assuming N to be degenerate and   to have a gamma distribution. The derivations which lead to the above are complex; readers are referred to Landberg and Singpurwalla (1985) for the details.

5. An Application: Optimal Testing of Software.

The failure models that have been reviewed in the preceding sections can be used for more than inference or the prediction of software failure. They can also be applied in the framework of decision theory to solve decision problems. An important example of such a problem is the optimal time to test software before releasing it. This involves the balancing of the costs of testing and the risk of software obsolescence with the cost of in-service failure, should a bug not be corrected during the testing period. The following is taken from Singpurwalla (1991), in which a strongly Bayesian approach is taken.

To implement a decision theoretic procedure requires two key ingredients. The first is a probability model, and here we take a generalization of the Jelinski & Moranda model. The second is a consideration of the costs and benefits, or *utilities*, associated with a particular decision i.e. the costs of testing, the benefits and costs of fixing a bug etc. Decision theory states that the optimal decision (in this case time of test) is that which *maximizes expected utility.*

If the software is to be tested for some time, say T units, and then released the problem is to find a T that maximizes expected utility. This is called *single stage testing*. There is a more complex, yet realistic, scenario called *two stage testing*. Here the software is tested for T units of time, and then depending on how many failures M(T) were observed during that test, a decision is made on whether to continue testing for a further T* units. The problem here is to find the optimal T and T*, with T* to be determined before M(T) is observed. Finally there is a third testing scenario, namely *sequential testing*. Here T* is determined after M(T) is observed; this procedure can continue for several stages, with T** being determined after M(T*) is observed and so on. Here we consider the case of single stage testing. Figure 3 is a graph of the decision process associated with single stage testing.

The model chosen in this paper is an extension to Jelinski & Moranda's model. We have

$$f_{T_i}(t \mid N, \Lambda) = \Lambda(N - i + 1)\, e^{-\Lambda(N-i+1)t}, \quad t \geq 0$$

In the previous section we placed prior distributions on one of N or Λ. Now we place priors on both the parameters, and say that N has a Poisson distribution with mean θ, Λ has a gamma distribution with scale μ and shape α and that N and Λ are independent.

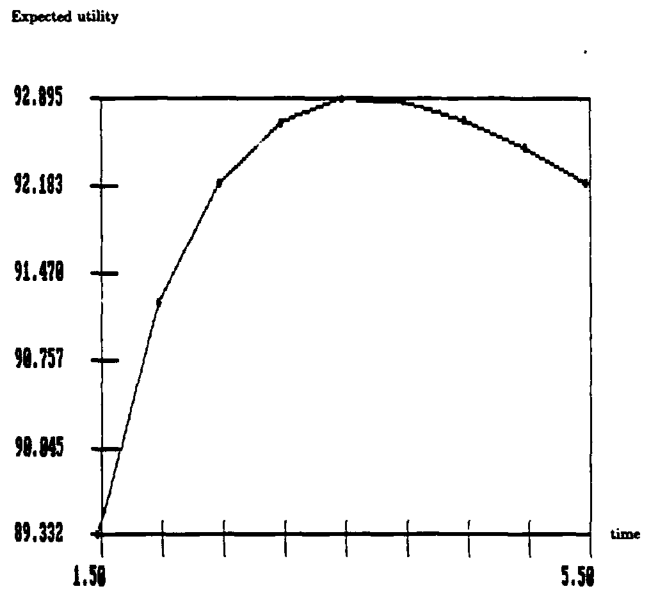
We now turn to the choice of utility function. The following assumptions are made:

i) The utility of a program that encounters j bugs during its operation is $a_1 + a_2 e^{-a_3 j}$.

ii) The cost of fixing a bug is some constant C_1.

Figure 4.

Time of testing versus expected utility for the software testing model.



iii) Let f(T) be the cost of testing and lost opportunity to time t; here we say $f(T) = dT^a$.

Note from i) that the utility of a bug-free program is a_1+a_2, and the utility of a program with a very large number of bugs is near a_1, so that typically a_1 is a large negative number (because there is a great loss associated with software that is constantly failing in the marketplace) and $a_1 > 0$. Combining these assumptions gives us the utility of a program that is tested for T units of time, during which M(T) bugs are found and corrected, and then released where j bugs are encountered by the customer as

$$U(T, M(T), j) = e^{-bT} \times [a_1 + a_2 e^{-a_3 j} - C_1 M(T) - dT^a]$$

where e^{-bT} is some devaluating factor.

Now the two parts of the decision process - the probability model and the utility function - are brought together. We wish to find the time $\hat{T}$ that maximizes expected utility. In other words find $\hat{T}$ such that $E(U(T, M(T), j))$ is a maximum, where we take expectation, using our failure model, with respect to M(T) and j. This maximization is quite complex, and must be done numerically via computer. The details are found in the paper, but the end result is best displayed as a graph of time against expected utility (figure 4); in this case one can see that the time one should test the software for is about 3.5 units.

6. Conclusion.

This paper has attempted to review the main methods, and some of the more well known models, that have been used by the statistics community in the area of software reliability. The first models were almost always based on looking at the failure rate of the software; later on the idea of modeling number of failures by a Poisson process was used and then most recently auto-regressive processes have been suggested as an alternative to the failure rate method. Application of the failure models, such as to the optimal testing decision problem, is another important aspect to the field.

Earlier it was pointed out that there is almost no data on the reliability of commercial software, due to the sensitive nature of that information. A possible method of overcoming this problem would be to have more interaction between the statistics and computer science communities. In the future, such interaction seems essential if models are to become more realistic and useful, and it is perhaps surprising that there are so few links between the two groups today.

There still remains much to be researched in this field. In the case of optimal testing, plans for two-stage and sequential testing need to be developed, whilst the verification of current and future models is likely to remain a problem. Nevertheless, because of the increasing presence of computers in all aspects of our daily lives, the topic of software reliability can only become more important in the future.

Acknowledgements

This report is based on a series of lectures given by the first author during the Fall of 1991 when he was the C. C. Garvin Endowed Visiting Professor of Computer Science and of Statistics at the Virginia Polytechnic Institute and State University. The comments of Professors I. J. Good, R. Krutchkoff, P. Palletas and K. Hinkelmann, and of students in the class, are gratefully acknowledged. We also thank Dr. Julia Abrahams of the Office of Naval Research for stimulating us to write this paper and for her comments on improving its readability. Also acknowledged are the inputs of Ms. Sylvia Campodonico and the assistance of Mr. Jingxian Chen.

Biographies

Nozer D. Singpurwalla is a Distinguished Research Professor of Operations Research and Professor of Statistics at The George Washington University, Washington, D.C. He has been a visiting Professor at Carnegie Mellon University, Stanford University, University of Florida, Tallahassee, University of California, Berkeley and Virginia Polytechnic Institute and State University. His current interests are in reliability theory, Bayesian inference, dynamic models and statistical aspects of software engineering. He has published over 100 papers and has co-authored a book on these subjects.

Simon P. Wilson is a Presidential Merit Fellow at The George Washington University, Washington, D.C. He has obtained a Bachelor's degree in Mathematics from Oxford University, England and a Master of Science in Operations Research from The George Washington University. He is currently enrolled in the doctoral program there.

References

1. Goel, A.L. (1983) A Guidebook for Software Reliability Assessment. Technical Report RADC-TR-83-176
2. Goel, A.L. & Okumoto, K. (1978) An Analysis of Recurrent Software Failures on a Real-time Control System. Proc. ACM Annu. Tech. Conf., ACM, Washington D.C., pp. 496-500
3. Goel A.L. & Okumoto, K. (1979) Time-dependent Error Detection Rate Model for Software Reliability and other Performance Measures. IEEE Transactions on Reliability, vol. R-28, pp. 206-211
4. Jelinski, Z. & Moranda, P. (1972) Software Reliability Research. Statistical Computer Performance Evaluation, W. Freiberger editor. New York: Academic, pp. 465-484
5. Landberg, N. and Singpurwalla, N.D. (1985) A Unification of Some Software Reliability Models. SIAM J. Sci. Stat. Comput., vol. 6, no, 3, pp. 781-790
6. Littlewood, B. (1980) A Bayesian Differential Debugging Model for Software Reliability. Proceedings of IEEE COMPSAC
7. Littlewood, B. & Verall, J.L. (1973) A Bayesian Reliability Growth Model for Computer Software. Applied Statistician, vol. 22, pp. 332-346
8. Mazzuchi, T.A. & Soyer, R. (1988) A Bayes Empirical-Bayes Model for Software Reliability. IEEE Transactions on Reliability, vol. 37, no. 2, pp. 248-254
9. Moranda, P.B. (1975) Prediction of Software Reliability and its Applications. Proceedings of the Annual Reliability and Maintainability Symposium, Washington D.C., pp. 327-332
10. Musa, J.D. & Okumoto, K. (1984) A Logarithmic Poisson Execution Time Model for Software Reliability Measurement. Proceedings of the 7th International Conference on Software Engineering, Orlando, Florida, pp. 230-237
11. Schick, G.J. & Wolverton, R.W. (1978) Assessment of Software Reliability, Proc. Oper. Res., Physica-Verlag, Wirzberg-Wien, pp. 395-422
12. Singpurwalla, N.D. (1991) Determining an Optimal Time for Testing and Debugging Software. IEEE Transactions on Software Engineering, vol. SE-17, no. 4, pp. 313-319
13. Singpurwalla, N.D. and Soyer, R. (1985) Assessing (Software) Reliability Growth Using a Random Coefficient Autoregressive Process and its Ramifications. IEEE Transactions on Software Engineering, vol. SE-11, no. 12, pp. 1456-1464